An Integrated Knowledge Engineering Approach To Process Modelling

Matthias Strickrodt

A thesis submitted in partial fulfilment of the requirements of the University of Glamorgan/Prifysgol Morgannwg for the degree of Doctor of Philosophy

This research project was carried out as part of a collaboration between the University of Glamorgan and the Fachhochschule Hannover

March 1997

Abstract

Industrial processes can often neither be modelled analytically nor identified experimentally with respect to all their nonlinear properties. For control engineering purposes, real processes are therefore approximated with models of relatively low complexity. Although such approximations are normally sufficient, it is often helpful to be aware of the variety of possible nonlinear effects to improve the design of control systems. Knowledge about these influences is frequently part of the industrial engineer's experience from working with the processes. At present, however, this experience can rarely be used systematically for control engineering purposes, since these process experts are in general not skilled in process modelling or control.

The subject of this work is the development of an approach that facilitates the systematic acquisition of the process expert's (or "area engineer's") experience in order to derive process models, which are possibly very nonlinear and/or multivariable or only partially defined, depending on the available information. This knowledge acquisition approach addresses therefore the above situation in that it serves as a new means of communicating the area engineer's process knowledge to the control experts. Furthermore, the area engineers themselves are able to use their experience in a systematic fashion for control engineering purposes. This latter aspect is addressed within a collaborative research project between the University of Glamorgan and the Fachhochschule Hannover, of which the work presented in this thesis forms the first part. Overall, the collaborative project aims at making Computer Aided Control System Design (CACSD) approaches accessible for engineers with little or no experience in control engineering.

A novel fuzzy hybrid approach for the simplified modelling of nonlinear multivariable dynamic processes is the first main contribution of this work. It builds on partial information about the global system behaviour in the form of locally valid singlevariable transfer functions. This important contribution forms an integral part of the knowledge acquisition approach to process modelling, the overall subject of this work. The systematic and structured knowledge acquisition procedure as such is the second major contribution. It facilitates the build-up, storage and flexible re-use of knowledge about static and dynamic system properties. In particular the above mentioned fuzzy hybrid models can be automatically generated through the prototype implementation of this knowledge engineering approach, without requiring any experience in modelling or fuzzy logic from the user. The third main contribution is a new modelling approach based on descriptive attributes referring to the process behaviour. This approach, which forms also an integral part of the knowledge acquisition procedure, facilitates the use of particularly abstract and not quantifiable information.

List Of Contents

List Of Figures

List Of Tables

Acknowledgements

I am most grateful to Mr. Keith Baker, the Director of Studies of this research project, for his support, advice and understanding throughout the project and for creating a motivating and inspiring atmosphere as well as for his successful efforts to obtain most of the funding for my research work.

Also, I would like to thank Prof. Dr. R. Schumann, the Second Supervisor, for his support in setting up this research project, his expertise and the challenging discussions throughout work, as well as his efforts to obtain the funding to cover the project-related travel expenses.

I would like to thank the University of Glamorgan, the Department of Mechanical and Manufacturing Engineering, the members of staff and especially Prof. J. Ward, my Third Supervisor and Head of the Department, for supporting my work - not least financially. Financial support from the British Council through the Anglo-German Academic Research Collaboration Programme, ARC, is also gratefully acknowledged.

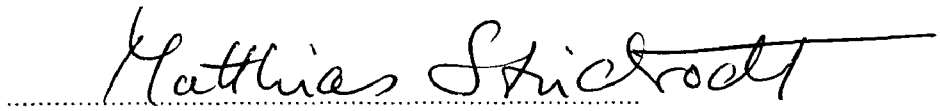
Furthermore, I would like to thank Steffen Körner and Birga Syska, the members of the collaborative research project, as well as Giuliano Premier for the discussions and feedback on my work.

For his early support in the arrangement of my contacts to the University of Glamorgan and the setup of my research project, I would like to thank Prof. Dr. Stannek.

Finally, I would like to thank particularly my family, especially my mother, for their support and encouragement. Last, but most certainly not least, very special thanks to my wife, Ana, for her understanding, patience, support and everything else.

Author's Declaration

This thesis has not been nor is currently being submitted for the award of any other degree or similar qualifications.



(Matthias Strickrodt)

Nomenclature

Symbol	Description
AI	Artificial Intelligence
a_i	polynomial denominator parameter of transfer functions G(s)
b_i	polynomial numerator parameter of transfer functions G(s)
CACSD	Computer Aided Control System Design
D	a) in short-hand for transfer function structure (e.g. "PDT1"): denoting derivative action b) as a parameter in a transfer function G(s): damping coefficient
fuzSC	short for "fuzzy static characteristic", according to the novel fuzzy hybrid approach proposed in this thesis
fuzTF	short for "fuzzy transfer function", according to the novel fuzzy hybrid approach proposed in this thesis
G(s)	time continuous transfer function
GUI	Graphical User Interface
I	in short-hand for transfer function structure (e.g. "PIDT1"): denoting integral action
KA	Knowledge Acquisition
KE	Knowledge Engineering
Kp	parameter in transfer function G(s): gain
LinAttrib	Class in the object structure of the proposed approach: Linear Attributes-based system description
LinMath	Class in the object structure of the proposed approach: Linear Mathematical system description
MIMO	Multiple Input / Multiple Output
MISO	Multiple Input / Single Output
MODEL^{ing}	the knowledge engineering approach proposed in this work
NL	non-linear, nonlinearity
NLd	nonlinear dynamic modelling sequence within the *MODEL*^{ing} approach
NLdAttrib	Class in the object structure of the proposed approach: Nonlinear Dynamic Attributes
NLdyn	Class in the object structure of the proposed approach: Nonlinear Dynamics
NLs	nonlinear static modelling sequence within the *MODEL*^{ing} approach

NLsAttrib	Class in the object structure of the proposed approach: Nonlinear Static Attributes
NLSC	Nonlinear Static Characteristic(s), also a class in the object structure of the proposed approach
NLstat	Class in the object structure of the proposed approach: Nonlinear Statics
OMT	Object Modeling Technique
OOD	Object Oriented Design
OP	Operating Point
P	in short-hand for transfer function structure (e.g. "PT1"): denoting proportional action
ProdRuleSet	Class in the object structure of the proposed approach: Set of Production Rules
RB..	Rule Base (".." denotes the parameter to which the rule base belongs)
SC	Static Characteristic
SISO	Single Input / Single Output
SISOdyn	Class in the object structure of the proposed approach: dynamic SISO systems
T	a) in short-hand for transfer function structure (e.g. "PDT1"): denoting lag b) as a parameter in a transfer function G(s): time constant
TF	Transfer Function
Tn	in short-hand for transfer function structure (e.g. "PTn"): denoting n-th order lag
U	process input, absolute
u	process input, relative
ΔU	input change (difference)
Y	process output, absolute
y	process output, relative
ΔY	output change (difference)

1. Introduction

Modelling is part of the everyday life of every human being. Without being aware of it, we build and adapt models of all sorts of things in our minds - from items as trivial as a water kettle to things as complex as our spouses. These models can be static ("with the toaster on setting 3, the bread comes out light brown; on setting 5, it comes out black") or dynamic ("it takes about 15 minutes for the oven to warm up to 200°C") and are often made up of a complex collection of experiences and hypotheses. Typically, process engineers in industry have a particularly good knowledge of the processes they work with because they can complement their hypotheses and detailed experiences of the static and dynamic process behaviour with theoretical understanding as well as numerical information.

1.1 The Subject Of This Work And The Need For It

The subject of this work is the development of an approach which enables these "area" engineers in industry to apply their process knowledge to build models that are appropriate for the further use and evaluation in the control engineering domain. There are two major reasons why the valuable experience of practitioners in industry should be systematically exploited:

1) In small or medium size companies, where control experts are usually not available (apart from high technology industry), such a modelling approach is needed for the introduction of systematic approaches to control system design, simulation and optimisation which can be handled by the area engineers.
2) In bigger companies, this modelling approach is needed as a means of communicating the area engineer's process knowledge to the control experts.

The former aspect addresses the absence of systematic control system design and optimisation approaches in big parts of manufacturing industry where intuition and rule of thumb tuning of control systems still prevail. Despite the trend of providing more user friendly interfaces for the established Computer Aided Control System Design (CACSD) environments, these are still not geared to the skill level of area engineers in industry but mainly aim at an improved handling for control experts. This is not to say that all CACSD programs should be simplified to a level that inexperienced users could handle them but that there is still scope for solutions that bridge the gap between such users and advanced technology. The resulting widening gap between advanced control engineering methods and the application in many industries is of major economic importance.

Aspect 2) on the other hand addresses the activation of a potential source of valuable practical information in companies with separate control engineering departments. So far, the control engineers in such corporate environments make only little use of the experience collected by the area engineers. The reason for this lies largely in the difficulties related to the translation of the practical experience into control relevant information. However, even the use of partially defined models which are derived from practical experience could significantly reduce costs and give new process insights, since theoretical process modelling is normally too time consuming and expensive for industrial applications. Although the practical experience of the area engineer should ideally be translated into a simplified but fully parameterised process model, the possibly only partially defined model could further be used as the important 'a priori' knowledge for improved process identification experiments.

For the development of practically applicable yet progressive approaches, it is especially important to understand the particular constraints of modelling and simulation in industry. Potential users of modelling approaches in most parts of manufacturing industry are relatively inexperienced in modelling and simulation, and in any case they are timewise extremely constrained. Therefore, the approach must be straight forward to handle without requiring a training and familiarisation phase.

More detailed information on the needs in industry that has been obtained through direct contacts and questionnaires is briefly summarised in the Appendix A1.

1.2 The Aims Of This Work

To address the above needs, the main objective of this research project was the development of a new, integrated approach to process modelling on the basis of knowledge and experience from area engineers. This approach had to be designed in such a way that its implementation into an interactive computer program was geared to the envisaged user group, the process experts, who have usually little or no experience in process modelling. Some particular aims of the approach are:

- to exploit different kinds of process knowledge
- to integrate various modelling approaches in a single approach
- to hide the complexity of modelling with the help of sensible default settings, where appropriate, as well as graphical user interface
- to guide the user through a self-explanatory modelling sequence, trying to avoid questions in the first place rather than answering any kind of queries

- to provide useful results to the other control system design modules, even if there is only partial process information available
- to integrate existing CACSD programs by generating appropriate simulation code in the case of sufficiently defined process models

In the overall context of Computer Aided Control System Design (CACSD), the suggested approach, called ***MODEL^{ing}***, will serve as a pre-processor to the modules 'Experimental Process Identification', 'Controller Design' and 'Process Simulation', which are being developed within the collaborative research project between the University of Glamorgan and the Fachhochschule Hannover. These modules will likewise be aimed at industrial users with only basic control engineering knowledge.

The term 'knowledge engineering approach' as part of the title for this research project is descriptive in the sense that the aim is largely to emulate the knowledge engineer who is, according to the understanding in artificial intelligence (AI), normally responsible for the acquisition and translation of the unstructured area expert's knowledge into a formal representation (here: a process model), as Figure 1-1 illustrates.

Although the *MODEL^{ing}* approach will not be able fully to replace a knowledge engineer, it could on the other hand even overcome some of the weaknesses of the knowledge engineer's interview techniques by avoiding the need for the area engineer to formulate all experience in words.

Apart from the aim of making modelling techniques accessible for engineers with little experience in process modelling, both control experts and engineering students should benefit from this approach, too. While the control expert is interested in obtaining some first, general simulation results through the application of such a quick, coarse and qualitative modelling facility, the engineering student needs to get a feel for the relationship between process behaviour and model representation.

Prerequisites for the applicability of such a knowledge-based modelling approach are the stability of the considered process and, for dynamics modelling, big enough time constants, so that the dynamic effects can be watched. These conditions are frequently fulfilled in the domain of process engineering.

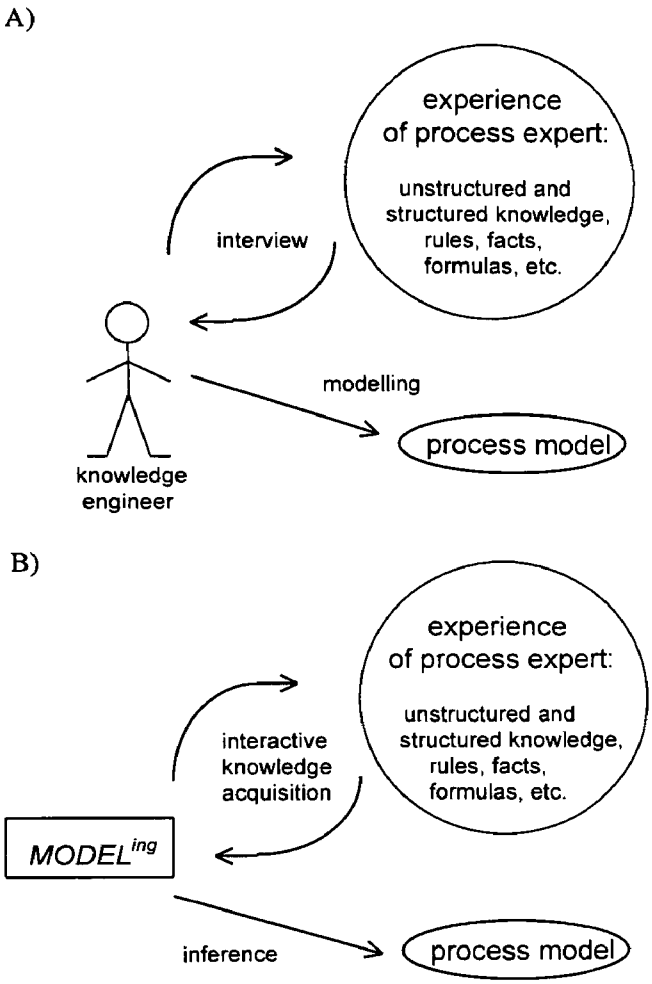


Figure 1-1: The Knowledge Engineering Approach

Knowledge engineers (A) are normally not available in industry. The interactive *MODEL*^{ing} program (B) should eventually do this job.

1.3 The Structure Of This Thesis

After the problem scope and need as well as the aim, application context and constraints have been described in the previous sections of this introductory Chapter, Chapter 2 gives a brief review of previous attempts to improve and simplify the non-trivial task of process modelling.

In the given context of this project, it was considered as particularly important to facilitate the modelling of nonlinear multivariable processes on the basis of incomplete - or "qualitative" - information. The emphasis of the literature research was therefore put on the review of qualitative modelling and simulation approaches (Chapter 3), in order to decide on which of these approaches might be suitable for the application within this work. Although fuzzy modelling turned out to be the most appropriate approach, it was found to be very limited with respect to the representation of

dynamics. Hence, a new fuzzy hybrid modelling approach has been developed which is particularly aimed at the modelling of nonlinear multivariable dynamic processes on the basis of only "patchy" knowledge. This new approach, which is introduced and validated in Chapter 4, has been specifically considered in the design of the knowledge acquisition sequence. Chapter 5 focuses on the layout of the *MODEL*^{ing} approach and is therefore a particularly important part of this thesis: it details the structure of the knowledge acquisition procedure together with the considerations that led to the particular design. In order to validate the *MODEL*^{ing} approach, it has been implemented in an interactive computer program. The implementational considerations are described in Chapter 6, together with the code generator, which facilitates the direct integration of the modelling results into a simulation environment. Additionally, the design of the self-explanatory graphical user interface, which is another important aspect of the *MODEL*^{ing} approach, is also described in Chapter 6. To validate the work, the computer implementations of *MODEL*^{ing} and the fuzzy hybrid approach have been tested. The results are summarised in Chapter 7.

Chapter 8 gives a general discussion of this work, its contributions and how it fits into the collaborative research project. It also summarises the applied research methodology and the results of the work. In Chapter 9, the final conclusions are drawn, before the possible further extensions of this work are detailed.

2. Modelling Support Approaches

The history of Computer Aided Control System Design (CACSD) started in the late 1960s and has developed substantially since [1, 2]. Although the abstract specification of the process to be controlled, i.e., the process modelling, is a prerequisite to the design and optimisation of control systems, there has been a significant 'lag', if not a 'dead time' with respect to the development of modelling support facilities. Despite important advances of the modelling support, it is still one of the areas of particular scope for further research within the field of CACSD. Some of the developments actually require a re-consideration: the intrinsic links between modelling and simulation that are currently the norm in commercial CACSD packages, for example, inhibit further developments towards multidisciplinary modelling. The most important concepts of modelling support, which have found different degrees of attention in the research community, are briefly summarised in the following sections. Experimental data-based identification approaches are, however, not considered in this overview since these are outside the scope of this work.

2.1 Graphical Modelling Support

One of the most important approaches to support the users of computer programs in general is the application of graphical user interfaces (GUI). In the domain of control engineering, this approach has been pursued by different research groups, most notably at the University College of Swansea [3, 4, 5, 6, 7, 8] and the University of Salford [9, 10]. With respect to process modelling, the basic idea is to offer a facility to define and manipulate systems graphically in a manner which is familiar to the user from the pencil-and-paper approach.

A variety of aspects in systems modelling can and should be addressed with the graphical user support facilities, such as:

a) construction of pictorial system representations (using specifically designed graphical editors)
b) input of textual system information (i.e., input of any alpha-numeric data, using text editors, form-filling approaches, etc.)
c) structural composition of systems from basic building blocks (hierarchical "bottom-up" modelling, combining subsystems to yield the overall model)
d) stepwise refinement - or decomposition - of a system structure (hierarchical "top-down" modelling)

e) choice of - often domain-specific - system descriptions (e.g., block diagram, signal flow graphs, bond diagrams, electrical circuit diagrams, mechanical symbol diagrams, Petri nets, process flow diagrams)
f) symbolic manipulation and transformation between different system representations
g) graphical verification facilities (e.g. consistency checks for structured system representations)
h) interface control features
i) standard "look and feel" and straightforward handling of all modelling steps

Although these aspects have been addressed in a variety of research projects, many of them are still not widespread and are therefore not readily available to users.

While purpose-built graphical editors (a), above) for the specification of topological system structure or functional relationships have become standard tools, simplifications for the textual input of systems information (b), above) beyond basic editors are not the norm. A very useful concept of textual input support, however, was presented in 1985 by Rimvall and Bomholt [1]: the command driven CACSD package IMPACT would switch either upon request or automatically, when detecting an error or missing information, into an interactive question-answer mode. This kind of interactive mode should be efficiently supported by the graphical user interface. By this means, especially matrix-based packages such as MATLAB could become a great deal more accessible for infrequent users.

Similarly, research projects like ECSTASY [2] and HIBLIZ [11] demonstrated some time ago the merits of providing both bottom-up (c), above) and top-down (d), above) modelling facilities, yet many of the standard tools still focus solely on bottom-up modelling.

The choice between different graphical system descriptions (e), above) is definitely not yet a standard feature either. A notable exception is the object-oriented modelling language Dymola [12, 13] with its graphical editor Dymodraw, which allows for domain-specific model views of different types. Even the combination of components in different graphical system description formats is possible, since the underlying definition is representation and application neutral. The possibility to import and export models in the neutral simulation model format 'DSblock' [14, 15] further enhances Dymola's suitability to multidisciplinary modelling.

Since different system representations are not the standard, neither are the transformations between representations (f), above). Via its neutral internal model format as well as by reading and writing to and from DSblock format, however, Dymola facilitates such transformations. The research environment eXCES includes algorithmic transformations between block diagrams and signal flow

graphs [5, 16]. Another example of the few existing transformation approaches is the 'MTT' toolbox [17], a research implementation based on symbolic computing, which transforms between bond graphs and different mathematical representations.

The direct *verification* of graphical system representations (g), above) is systematically implemented in the object-oriented approaches Omola [18, 19, 20] and Dymola, where the notion of "across" and "through" variables plays an essential role in ensuring consistency right from the start. Following the actual modelling process, *external validation*, however, is mostly featured, using the simulation facilities of the CACSD tool.

Interface control features (h), above) include facilities to zoom, scroll, move, open and close - or generally to manipulate - components of the GUI according to the user's needs and skills. Such features are standard parts of most modern control engineering environments.

"Look and feel" is mainly concerned with the appearance and behaviour of an application. Standard "look and feel" concepts (i), above) have largely been implemented within the current CACSD packages, but there is not yet a standard across the different implementations. Increasing convergence towards the Microsoft Windows style, however, is to be expected. On the other hand, "Look and feel" in the sense of visual guidance that enables the uninitiated or infrequent user to feel his/her way forward in applications as complex as CACSD is virtually non-existent. This is despite the fact that projects such as GE-MEAD [21], where the interface and level of guidance adapts to the user's needs and skills, have broken some ground in this respect.

Overall, the aspects (a) to (g) can be summarised under *application specific* considerations, whereas aspects (h) and (i) address also more general requirements for GUI. Graphical interface design and functionality is an issue in itself, yet it is very closely linked to the object-oriented paradigm [22, 23] which enables many developments that would be unthinkable on the basis of conventional approaches - not least because of complexity and maintenance issues.

2.2 Model Management

The management of previously modelled processes is an important part of powerful editing and model re-use facilities which play a key role in the reduction of the modelling workload. Model management is mainly achieved through *version control*, *component libraries* and *browsers* in conjunction with a database system.

Version control mechanisms have to ensure that updated and previous versions of a model can exist concurrently so that the modeller can fall back on any intermediate status during the lifetime of the project. *Component libraries*, containing the most frequently used domain-specific sub-models and basic building blocks are essential for efficient model aggregation. *Browsers*, finally, must support a variety of search paths to narrow the choice of models in the libraries for any specific application. Furthermore, an ideal multiple window browser gives already a preview of the models and indicates the application contexts.

In addition to handling the actual process models, however, this support facility should be extended to a project management tool in the overall context of CACSD [24]. This latter level of support includes the documentation of, for example, modelling assumptions, configurations and conditions, changes made to the model and the reasons for the changes as well as controller structures, parametrisations and simulation results.

Here, too, the way forward is object orientation with the powerful notion of inheritance: In particular object-oriented database management systems hold promise [23]. While libraries of basic building blocks are already the standard in CACSD, the systematic re-use of component structures is only well formalised in Omola and Dymola. However, the actual 'management' aspects in modelling depend for the time being mostly on the user's own initiative and determination. Overall, the research environments ANDECS [25, 26] and GE-MEAD [21] appear to feature the most sophisticated project and model management facilities and are therefore good 'landmarks' for more widespread developments. The suggestion of an integrated approach to project management in computer aided control engineering by Barker et al. [24] goes even beyond the functionality of ANDECS and GE-MEAD and stresses the importance of a neutral data model.

2.3 Symbolic Computing

Symbolic computing can be applied to alleviate the burden of system modelling by offering different services. It can actively support the model construction and simplification, it can be used for "translating" between various graphical and textual system representations and it is applicable to the analysis of process models.

This application of symbolic computing as a wide-ranging modelling support facility lies still some way ahead in the future, although such prospects have already been confirmed by research implementations that cover some of the aspects: the determination of and transformation between

plant model representations using symbolic computing is featured in CES [5] and the 'MTT' toolbox [17], respectively.

Existing symbolic algebra packages like Mathematica, MACSYMA, or the MATLAB Symbolic Toolbox could efficiently be applied for further developments in this direction.

2.4 Intelligent Help Systems And Expert System Support

Since the user guidance and help facilities could not keep pace with the increasingly complex CACSD environments, these programs (including their modelling tools) moved further and further out of reach for uninitiated and infrequent users. Two approaches to overcome this problem emerged from Artificial Intelligence (AI) [27]:

1. To substitute the conventional interactive help facilities with an information retrieval system (intelligent help) giving the user the impression of customised guidance.
2. The introduction of an extra front end in the form of an expert system guiding the user through the performed operations.

The former technique is still fairly passive in a sense, in that it leaves all major execution control to the user. Different types of intelligent help systems have been developed which can be subdivided mainly according to the type of "user modelling", the event triggering the activation of the help function and the behaviour after activation [28]:

- In intelligent help systems, an internal model of the user and his/her actions is often built. This 'spying on the user' is done in order to assist context sensitively upon request or occurrence of mistakes. In addition, the mode of dialogue can be changed in accordance with the user skill level. Such systems differentiate between explicit querying of the user and implicit modelling based upon a constant monitoring of user actions.
- The help function can be triggered actively via the user model when the need is automatically recognised or passively upon request of the user.
- The behaviour after activation is differentiated by Hoffmann and Rimvall [29] between a passive help mode (which provides specific information on help topics), a query mode (in which commands are checked interactively for completeness and correctness), and finally an extended guiding mode (which is defined as a mixed informational-executional mode).

The decision as to which combination of the latter modes of intelligent help systems should be implemented is dependent upon both the category of users to be served and upon the complexity of the software to be supported. To define the right recipe for the help system is, in fact, a major difficulty because usually no two users, even within one 'category', share the same needs of support.

Although the idea of introducing the user via intelligent help systems to command driven matrix environments seems to be suitable, the general problems of industrial area engineers are not addressed. This is because the time consuming introduction to command driven systems does not pay off for the industrial engineer who normally uses a CACSD package less frequently - even if this engineer were granted the time to get a thorough understanding of the program, he/she would have to refresh his/her knowledge whenever he/she had to work with the package.

Unlike an intelligent help system, an expert system takes over the key role in the man-machine relationship - it leads the user through the whole problem solving procedure or solves complex problems outright.

Expert systems essentially consist of a big rule base which should provide specific decisions with respect to the procedure and actions to be taken in any occurring situation. This idea, however, leads in a field as wide as control theory to an enormous rule base without ever being able to cover every situation and problem. Therefore, the most common application of expert systems is the so-called 'narrow-deep' approach which supports the solving of specific complex problems. In order to be able to cope with the variety of practical problems in industry, however, a 'broad-shallow' approach is more appropriate.

Since the guidance given by expert systems can easily be too strict [1], care has to be taken in the design of such systems so that the user remains in control of the overall process. If, however, particular guidance is required in parts of the modelling approach, measures should be taken to avoid the situation in which the user feels relegated to an information supplier by:

- keeping the user updated by informing on the status of the interactive process (for example, the user should see the effects of the information immediately)
- providing a tracking facility so that the user can follow the complete decision process after completion
- speeding up even complex processes significantly so that the tool is attractive even for experienced modellers.

Meier zu Farwig and Unbehauen [30] distinguish between the terms 'consulting system' and 'expert system': "A consulting system represents a technical system that on the basis of appropriate knowledge types supports human beings in decision finding ... by decision proposals." and "An expert system ... is a special type or a part of a consulting system that supports the user on the basis of declarative knowledge and an inference strategy in decision finding." Although consulting systems generally leave more responsibility with the user, expert systems do not have to be patronising either, as the expert system shell for control system design within the GE-MEAD environment [21] shows.

To sum up, the amount as well as the type of artificial intelligence (AI) employed in a CACSD program has to be carefully selected. Taylor [31] details the considerations on this issue further. Without depriving the user of responsibility for the overall problem solving procedure, 'local' AI-support can in fact enable approaches which are otherwise impossible.

2.5 Intelligent Modelling Approaches

As opposed to 'ordinary' expert system support for modelling which aims to help the user cope with the complexity of standard modelling approaches, intelligent modelling goes further in that it focuses on the direct incorporation of the modeller's process knowledge into an appropriate knowledge representation in the form of a suitable process model. With the modelling knowledge implemented in the intelligent modelling approach, the user supplies only his/her knowledge about the process in a convenient format. The modelling system is responsible for the translation of the process knowledge into the actual process model and simulation code.

Collecting and storing the process information, the intelligent modelling approach acts therefore as a "knowledge accumulator". Obviously, the consideration of an appropriate (process-) knowledge representation, which is independent of any particular simulation environment is therefore an important issue. The "object-oriented information model for intelligent modelling", suggested by Li, Jobling and Grant [32, 33] is particularly interesting in this respect: being based on the object modelling technique (OMT)[1] by Rumbaugh et al. [34], it takes advantage of the similarities between object-oriented system modelling and the design of object-oriented software.

Particularly noteworthy work in the field of intelligent modelling has been carried out at the University of Sheffield between 1987 and 1993 [35, 36, 37]. The 'Knowledge-based Environment for Modelling and Simulation' (KEMS) focuses in its modelling part on the user's structural analysis of

[1] OMT is described in more detail in Chapter 5.

the process and the synthesis of an appropriate representation on the basis of standard components. Prior to this main application, the system's underlying 'deep' knowledge[2] (e.g. physical laws) is implemented by a modelling expert, using a 'Knowledge Acquisition Module' (KAM) [38, 39, 40]. Additionally, a model validation module checks automatically the integrity of component interconnections and supports the user in the analysis of simulation results [41].

A very different, yet interesting approach is the use of a "sublanguage", a bounded subset of the English language, for the direct definition of dynamic systems [42]. The drawback of such approaches, however, is the likelihood of misunderstandings between the user and the system which calls for very complex parsing systems. Furthermore, the model specification requires extensive typing and experience in modelling.

Despite the above mentioned conceptual distinctions, there is indeed a close relation between intelligent modelling and expert systems, as the similarity between KEMS and GE-MEAD for example shows. However, although the term "intelligent" is nowadays frequently associated with knowledge bases and expert systems, it can just as well refer to any other approach that represents the domain knowledge. In fact, conventional approaches like decision trees are still often advantageous to rule-based approaches, depending on the degree of structure in the knowledge [43].

2.6 Knowledge Acquisition Approaches

Knowledge acquisition mainly refers to the activity of "knowledge engineers", who acquire and formalise knowledge by reading textbooks and in particular by communicating with the domain specialists for the purpose of developing knowledge based systems [44]. Further to this *indirect* -or *deductive* - knowledge acquisition approach, the more efficient *direct* - or *inductive* - knowledge acquisition, in which the domain specialist directly interacts with a knowledge acquisition program that takes the role of the knowledge engineer, is gaining increasing importance in the field of Artificial Intelligence (AI) [45, 46]. A third type, the *automatic* knowledge acquisition, aims at the extraction of knowledge from process data [46, 47].

The term "knowledge acquisition" has, however, generally been used in quite an undistinguished manner. General expert system tools or even universal programming languages are, for example, mentioned in the context of direct knowledge acquisition tools [45]. It should therefore be stressed that the word "acquisition" implies an *active* role, whereas a system - or a person - that *passively*

[2] The different knowledge types are defined in Appendix A2.

awaits the provision of something (knowledge, for example) is at best collecting or accumulating, rather than acquiring. An example in the context of process modelling is the Knowledge *Acquisition* Module (KAM) of the previously discussed KEMS environment, which would have to be labelled more correctly as Knowledge *Collection* Module: similarly to the knowledge engineer, who (inter-) actively acquires knowledge from the process expert by asking well structured questions, an acquisition module, which takes the knowledge engineer's role, must have 'an idea' of what kind of information to look for and then try to actively *acquire* it. Awaiting passively the user's input of code in the high-level language 'FKRL', KAM merely generates Prolog code and thereby increments the knowledge base [37, 40]. With the introduction of a graphical form-filling approach [40, 48], the role of KAM in the knowledge build-up became only slightly more active [49]. Nevertheless, the relatively passive character of KAM suits its purpose very well since it is aimed at the modelling expert rather than the process domain specialist.

Most *direct* and *indirect* knowledge acquisition approaches in the domain of control engineering refer to the acquisition of process operation and control strategies from the process operator or domain specialist. In view of the difficulties with the indirect acquisition of process operator's strategies [44, 50], the direct acquisition appears to be far more promising [46]. Formulated in production rules and implemented in fuzzy controllers or expert systems, this knowledge is directly applied to the control of the process [44, 46], without the need for a mathematical model.

In the wider sense, process identification approaches could be considered as *automatic* knowledge acquisition approaches. Isermann [51] and Eykhoff [52] give comprehensive introductions to conventional identification approaches, which are in parts supported by CACSD programs. For an overview of more recent approaches that are based on connectionistic structures, please refer to [53]. A very interesting approach is "ROSA" [47], which automatically generates production rules on the basis of process data. As was mentioned earlier in this chapter, the data based (i.e., "*automatic*") approaches are not considered here in more depth as they are outside the scope of this work.

In the context of computer-based modelling support, which is the subject of this overview, only the *direct* knowledge acquisition for process modelling is therefore of interest here. Such approaches are useful, for example, for simulation purposes, optimisation of conventional control systems or as a means of communication of process experience. This area has, however, found very little attention in the research community so far, with the work by Sawaragi and Nakamori [54, 55], which focused on the acquisition of quite coarse information to eliminate unimportant cause-effect variable pairings, being an exception. Obviously, this area is closely related to the issue of intelligent modelling, but it puts more emphasis on the active role of the program in the system-user

interaction. This active role is typically mirrored by a procedural design of the *direct* knowledge acquisition program [46].

2.7 This Project In Relation To The Previous Work

Without being all-inclusive, the above overview of different "branches" of modelling support gave a broad idea of the main areas, some of the work that has been done and the further potential. Obviously, these areas are not to be considered as strictly distinct fields but there exists a significant amount of overlap. Although browsers are, for example, important parts of the model management, they are likewise part of the interactive graphical user interface.

In general, the modelling simplifications that have so far been introduced in both commercial and research implementations are aimed at making control engineering programs more accessible and easy to use for the control engineer. The control engineering expert in industry has generally been identified as the main beneficiary of these improvements. *Without* the requirement of both extensive programming skills and a substantial amount of time for low level interactions, the modelling support facilities enable the user to focus on his/her main tasks - the development of process models for control system design and simulation.

Apart from the individual areas of scope for further improvements to the modelling support that have been indicated in the above sections, there is an overall issue that has found very little attention: the design of support facilities for potential users that require even more support than the industrial control engineer. 'Area engineers', mostly with a chemical or manufacturing engineering background, whose main task is the supervision and optimisation of production processes are normally also confronted with control engineering issues, although on a less frequent and somewhat 'peripheral' basis. In large parts of industry where specific control engineering departments are non-existent, the control system design and optimisation is even fully in the hands of such area engineers. In the light of present day requirements, there is a strong economical and ecological need to replace the prevailing 'rule of thumb' approach in these industries with more systematic methodologies. As part of a collaborative project between the University of Glamorgan and the Fachhochschule Hannover which aimed to address this need with a purpose-built approach to CACSD for industrial practitioners, this thesis focuses on a specific part of process modelling.

The only previously reported work on modelling support that considered explicitly the area engineer as one of the potential user types is the 'Knowledge-based Environment for Modelling and Simulation' (KEMS) that was developed at the University of Sheffield [48, 49]. In KEMS, the area

engineer is provided with a library of components and submodels which are used for the graphical design of the topological system structure. The components are pre-defined in terms of their underlying physical equations - the 'deep' system knowledge. For the input of component-specific information such as parameters, a form-filling approach is taken. The task of creating new component models, however, was considered to be an expert modeller's task which would be carried out infrequently [48, 36], using the separate program module 'KAM'.

Similarly to KAM, *MODEL*^{ing} , the approach described in this work, focuses on the specification of components that have not been previously defined. Different to the purpose of KAM, however, it is mainly aimed at area engineers, and should allow these users to apply their process knowledge to modelling without having to fall back on specialist modellers for any part of the approach. *MODEL*^{ing} aims to consider different aspects of modelling support that have been discussed in the above sections, but clearly focuses on the concept of knowledge acquisition with the help of an appropriate graphical user interface. More specifically, *MODEL*^{ing} is a *direct* (or *inductive*) knowledge acquisition approach for process modelling. The development of the structured procedural, yet flexible approach is therefore of central importance for this work as it ensures the active role of the system in its interaction with the area engineer. The self-explanatory GUI plays an important role, because the concept of avoiding questions altogether (or as far as possible) is clearly advantageous to providing a query system which can offer information on any arising question.

'*MODEL*^{ing}' is primarily the new modelling approach, with its prototype implementation in a computer program serving the purpose of validating the approach. The overall idea of this work is to elicit, build-up and store as much useful process knowledge as possible. Therefore, the results of applying the *MODEL*^{ing} approach will vary between coarse structural models and fully parametrised transfer functions, depending on the amount and type of knowledge that is available. A review of qualitative modelling and simulation approaches, which deal with such incomplete process knowledge is summarised in the following chapter.

3. Survey And Evaluation Of Qualitative Modelling And Simulation Approaches

In the context of this work on knowledge acquisition for process modelling, it is quite likely that the available process knowledge takes different forms, is 'patchy' and incomplete. To decide on the way of utilising this partial, qualitative knowledge for modelling and simulation, the following review had to be carried out[1].

Qualitative modelling and simulation is an extensive field of research which is quite difficult to review for someone who mainly wishes to apply a modelling technique that allows for incomplete system knowledge. Avoiding detailed descriptions of the algorithms, this survey refers to the further literature and focuses on the characteristics that are of particular importance in control and knowledge engineering. Firstly, however, attempts to clarify the role of qualitative approaches in the context of modelling in general are discussed and categories for grouping the reviewed approaches are defined.

3.1 Qualitative Process Modelling

Researchers from different fields - like process and control engineering, maths, computing, physics and artificial intelligence - have been working on 'qualitative modelling'. Due to the different backgrounds, misunderstandings and contradictory definitions are fairly commonplace. Therefore, Lunze [56] stated quite rightly in 1992 that there was not yet a clear and general definition of a qualitative model.

Quantitative or 'crisp' models (e.g. fully parameterised differential equations or transfer functions) are either not available or of little use if one of the following circumstances applies:

- The dynamic system is not completely known with respect to the structure of the underlying differential equation or its parameter values.
- The actuator signals of complex plants are manually adjusted by an operator who takes various indicators of the current process state into account.
- Only coarse measured signals (e.g. high / medium / low) or indirect process state indicators are available.

[1] This survey has been submitted for publication in the IEEE Trans. on Systems, Man, and Cybernetics. The manuscript is largely identical to this chapter and is appended to this thesis for reference.

Although this specification of the typical situations in which quantitative modelling is inapplicable and therefore qualitative or semi-quantitative modelling becomes a necessity is expressed in control engineering terms, it applies accordingly to all other strands of research. Based on the analysis of these typical situations, Lunze [57] suggested in 1993 his own definition:

> "Models, which are based on a coarse evaluation of signal and parameter values are denoted as 'qualitative models'. These models often refer to symbolic instead of numeric values with respect to the signals and their parameters." Lunze [57]

As opposed to this open and general definition, many researchers (like DeKleer-Brown [58] and Forbus [59]) consider 'qualitative modelling' as the generic term for all methodologies which use only the signs of parameters and influences.

In this survey, however, qualitative techniques in the latter sense are considered together with 'semi-quantitative' approaches that also use some sort of quantitative information. The view that any model which is not fully defined in a quantitative way is somewhat 'qualitative' - with varying degrees of quantitativeness - is therefore shared here (Fishwick [60]).

The few publications on control engineering applications of qualitative modelling mainly focus on the third of the above cited circumstances, the availability of only coarse measurements. This survey, however, emphasises the knowledge engineering point of view and therefore focuses on the first two aspects, which refer to the incomplete knowledge of a process.

The Role Of Qualitative Models In The General Modelling Context

To clarify the role of qualitative models within the general modelling context, it is useful to subdivide qualitative models according to their degree of quantitativeness. In section 3.3, the different categories of qualitative models will be important for the discussion of the properties of the modelling and simulation approaches.

Different suggestions for such model categories have been made. Stevens et al. [61] categorised models hierarchically as follows:

- *Structural models* - containing only cause-effect and connectivity information
- *Qualitative models* - like structural models with additional qualitative dynamic information
- *Static models* - like structural models with additional numerical steady-state information

- *Quantitative models* - containing all the structural and numerical dynamic information about a system such that it is theoretically possible to obtain a complete and accurate description of the system's behaviour

These categories, however, are not truly hierarchical as the information in static models is not a superset of the information covered by qualitative models. A more consistent suggestion was made by Mavrovouniotis and Stephanopoulos [62]:

- *Boolean models* represent only the existence of parameters and interrelations between them without any information on signs or magnitudes.
- Additionally, *qualitative models* represent the signs of variables and the direction in which each variable affects another one. Information on magnitudes or relative orders of magnitudes is not provided. A signed, directed graph (digraph) showing how parameters affect each other is an example of this type of model.
- *Order-of-magnitude models* provide, in addition to the information covered by qualitative models, some rough (absolute or relative) magnitudes of parameters and effects.
- *Quantitative models* employ the most detailed numerical and algebraic representations, such as systems of equations and numerical values of parameters.

This hierarchy is consistent but it is - like Stevens' model hierarchy - not complete for two reasons. Firstly, because *heuristics*[2] based models, which play an important role in qualitative modelling, are not considered and secondly, because conventional quantitative models are not necessarily the best - or most detailed - abstraction of the real process. Mavrovouniotis himself corrected very recently [63] his old concept with respect to the most exact model and stated: "it is not crisp values that represent complete knowledge, since only one specific system in a specific state can be described by crisp values". He suggests lumping together many crisp descriptions in a "joint distribution function" for all the variables of a system as the most detailed form of knowledge.

It is, however, not possible to allocate *heuristics* based models appropriately within this model hierarchy since their degree of quantitativeness can vary significantly and they can cover information on the modelled process beyond the scope of quantitative models. A strictly hierarchical concept does therefore not appear to be an applicable format to order the different types of process models, unless the scope of the model ordering concept is explicitly restricted to *causal*[3] models. In

[2] Heuristic knowledge is knowledge which points from problem features to problem solutions and can be formulated in if-then rules.

[3] Causal knowledge is knowledge about general relationships between problem solutions and problem features [45]. In the context of qualitative modelling it refers to links between events and states. Causal knowledge is mostly formulated in mathematical equations and relationships.

this case, neither *heuristics* based models nor the "joint distribution function", which represents statistical knowledge, have to be considered.

An appropriate hierarchy of ***causal*** process models is therefore:

- ***Boolean or structural models*** represent only the existence of parameters and interrelations between them without any information on signs or magnitudes.
  example: a second order proportional transfer function
  $G(s) = b_0 / (a_2 s^2 + a_1 s + 1)$ with $a_i, b_i \neq 0$.
- Additionally, ***sign-based qualitative causal models*** represent the signs of variables and the direction in which each variable affects another one. Any quantitative information is abstracted to the set { -, 0, +}
  example: transfer function $G(s) = [-] / ([+] s^2 + [+] s + 1)$.
- ***Semi-quantitative causal models*** provide, in addition to the information covered by sign-based qualitative models, some rough (absolute or relative) magnitudes of parameters and effects. The semi-quantitative information can be represented by inequality relations (e.g. $a_2 >> a_1 > 0 >> b_0 \geq -100$), value ranges (e.g. $a_1 = [0.8 \ldots 13]$), fuzzy sets or other formalisms.
  example: transfer function $G(s) = [-50 \ldots -100] / ([20 \ldots 150] s^2 + [0.8 \ldots 13] s + 1)$.
- ***Quantitative models*** employ the most detailed numerical and algebraic representations, such as systems of equations and numerical values of parameters.
  example: transfer function $G(s) = -66 / (134.6 s^2 + 7.2 s + 1)$.

This hierarchy is in accordance with Mavrovouniotis and Stephanopoulos [62], but more specific and generally applicable as far as the labels of level 2 and 3, respectively, are concerned.

Boolean models - the most abstract qualitative models - are very useful as 'a priori' information for identification experiments as well as the structural layout of a control system but cannot be used as such for simulation purposes. Purely structural models are therefore not considered in the following discussions on modelling and simulation characteristics. Nevertheless, their importance is not neglected since the structural models are a sub-set of all other causal models. Since the fully defined quantitative models are not the subject of this overview, they will not be considered any further either.

In section 3.3, the remaining two categories of the above *causal* model hierarchy (***sign-based qualitative causal models*** and ***semi-quantitative causal models***) are complemented by ***heuristics based semi-quantitative models*** to form the three distinct categories in which the surveyed approaches are sorted.

3.2 Overview Of The Different Approaches

In this section, the different modelling and simulation approaches as well as their basic concepts are very briefly introduced.

The publication of Hayes' paper 'The Naive Physics Manifesto' [64] in 1979, in which he proposed constructing a formalisation of a large part of ordinary everyday knowledge of the physical world, was of key importance for the developments in qualitative reasoning. Hayes criticism was that problem solver programs in Artificial Intelligence (AI) addressed only 'toy-problems' and he further detailed the characteristics of the required formalism. Several research projects were triggered off by his ideas which were aiming at the automation of the techniques by which humans reason about the physical world on the basis of very general - or 'qualitative' - information. The following approaches which appear here without a strict order are mostly - either directly or indirectly - inspired by Hayes' concept.

The **'Qualitative Physics based on Confluences'** was developed by De Kleer and Brown [58]. In this theory, qualitative constraints are associated with the components and connections that make up a mechanism. The formalism is based on the concept of confluences, i.e., qualitative differential equations with parameter values abstracted to '-', '0' or '+'. The simulation result is a directed graph of qualitative states that corresponds to the set of all possible sequences of events that can occur from the initial qualitative state ('envisioning', Figure 3-1).

The **'Qualitative Algebra Q1'** by Williams [65] is an extension of De Kleer and Brown's concept and therefore also based on the availability of mathematical model equations. In Q1, more algebraic operators are allowed than in Qualitative Physics and the abstraction to the qualitative parameters {-, 0, +} is made at a later stage, so that the equations are initially operated on using a symbolic algebra system. Yielding a reduced number of possible future states, the ambiguity of the predictions can be reduced with Q1.

The **'Quantity Lattice'** was designed by Simmons [66] specifically to handle problems with thousands of variables, expressions and inequalities in a computationally efficient manner. The idea was to trade completeness of information for faster, more intuitive deductions. Similarly to the first two approaches, the relationships of parameters need to be specified in mathematical format. In the Quantity Lattice, both simple arithmetic expressions (+,*,-,/) and ordinal relationships ($>,<,=,\neq,\geq,\leq$) are supported.

Forbus' **'Qualitative Process Theory'** [59], which was implemented in **'QPE'** [67], has partly evolved from De Kleer and Brown's work and uses the same concept of confluences and envisioning. Unlike Qualitative Physics, however, Forbus' approach concentrates on modelling physical *processes* (for example flow from A to B) rather than on modelling *components and their interconnections* (like tanks which are connected through pipes). These processes are activated when the appropriate conditions exist and individuals, i.e. physical objects, are present.

Extensions to the Qualitative Process Theory that aimed at enhancing the feature of modularity of this approach have been developed by Falkenhainer and Forbus [68]. To compose a model from different modules, the approach requires a 'domain model', i.e., a library that consists of 'fragments', each describing some fundamental piece of the domain's physics. The user specifies a 'scenario' (important objects, initial conditions and relations) and the simulation environment instantiates the appropriate fragments to yield the scenario model before the simulation is run. A central idea is the explicit statement of the modelling assumptions. The system has therefore been applied in conjunction with numerical simulators (e.g. Runge-Kutta) to monitor simultaneously whether any of the modelling assumptions are violated during the numerical simulation. **'SIMGEN'** by Forbus and Falkenhainer [69] requires additionally a mathematical model library that corresponds to the 'domain model'. SIMGEN is used to generate automatically a numerical model from the qualitative as well as semi-quantitative information (scenario). The main idea of SIMGEN is to give explanations and to generate answers to questions about the trajectory of the numerically simulated model. Unlike in NSIM (see below), the automatically generated numerical model is an exact model that does not express the inexactness of the qualitative model in any way.

'Qualitative Process Theory using linguistic Variables' by D'Ambrosio [70, 71] is an extension of Forbus' theory which aimed at overcoming the "severe limitations of QP theory" [71] (i.e., mainly the ambiguity of simulation results) by combining it with the fuzzy theory notion of linguistic variables. Quantitative and semi-quantitative information can therefore be handled to a certain extent in this extension to Qualitative Process Theory.

The **Qualitative Simulation** Approach **'QSIM'** by Kuipers [72, 73, 74, 75, 76] is - like all the above listed formalisms - based on the availability of information about the mathematical relationships between the process variables. Quantitative information is abstracted to {-, 0, +} but ordinal relations with some 'landmark values' are included additionally. Although Kuipers also used the 'envisioning' concept suggested by De Kleer, QSIM features improved facilities to reason about time compared with the other confluences based approaches.

Over the years, QSIM has been extended significantly towards using (semi-) quantitative knowledge: While **'Q2'** by Kuipers and Berleant [77] assigns relatively coarse value ranges to previously labelled landmark values, **'Q3'** by Berleant and Kuipers [78] with its temporal step-size refinement is a significant step towards combining the strengths of both qualitative and quantitative simulation. In Q3, different degrees of semi-quantitative information can be used, with the output quality being directly related to the quality of the information provided. Unlike Q3, which is concerned with the interpolation of the behaviour at discrete time points, **'NSIM'** (Kay, Kuipers [79]), a third extension to QSIM has specifically been built to increase the precision over the intervals between the time points. NSIM generates for each initial state of the semi-quantitative, qualitative differential equation a set of numerical - normally nonlinear - extremal equations which bound the state variables, and in a second step applies any numerical simulator, such as Runge-Kutta, in order to generate the 'dynamic envelopes', i.e., the trajectories which bound the possible behaviours of the semi-quantitatively defined system. NSIM and Q3 each have advantages and disadvantages with respect to each other and are therefore complimentary approaches that both build on Q2 which in turn builds on QSIM.

'QPC', developed by Farquhar [80], Crawford and Kuipers [81], synthesises the advantages of both the 'Qualitative Process' approach by Forbus [59] and 'QSIM' by Kuipers [75]. It applies and extends the concept of the former expressive compositional model building approach and generates the set of initial conditions suitable for solution by the qualitative simulation engine of the latter approach. The main concept of this compositional modelling approach is that the individual situation is only to be specified in terms of the 'scenario', i.e., objects that are known to be of interest, some initial conditions, and some relations that hold throughout the simulation. Using an appropriately pre-specified 'domain theory' , i.e., a model fragment library that contains for example physical laws (like mass conservation), processes (like liquid flows), devices (like pumps) and objects (e.g. containers), QPC activates automatically the model fragments whose conditions are satisfied either by the initial 'scenario' specification or at any time step during the simulation (therefore 'compositional'). On the basis of the active model fragments, QPC generates constraints that are translated into qualitative differential equations (QDE) as inputs to QSIM and, after computing an initial state, triggers directly the QSIM simulation run.

'SQPC' (Farquhar, Brajnik [82]), the semi-quantitative extension to QPC, handles additionally numeric bounds on magnitudes and monotone functions, functions specified by look-up tables and dimensional information, which enables very helpful consistency checks. For simulation, it applies QSIM with its semi-quantitative extension Q2; a further extension towards using additionally NSIM is currently being developed.

The **'Order-of-Magnitude Reasoning - O(M) -'** approach by Mavrovouniotis and Stephanopoulos [62] / Mavrovouniotis [63] is different from the others in that it was particularly designed for static modelling for engineering purposes with the facility to include semi-quantitative information. To define the orders of magnitudes that hold amongst the system variables and/or the values of the variables, seven primitive relations such as "much smaller than" and "moderately larger than" are applied. Each of these relations is interpreted with respect to the location of the quotient of the two compared quantities within an interval. All such intervals - which are disjoint - are defined with respect to a unique parameter chosen according to domain knowledge. The possibility of making efficient use of higher-quality knowledge - possibly even numerical parameter and relationship information - is, however, limited. One of the interesting characteristics of this approach is the possibility to combine a system's mathematical relations with rule-based descriptions. Also, several different relations between two quantities are allowed to coexist.

'Formal Order-of-Magnitude Reasoning, FOG' developed by Raiman [83] is similar to Mavrovouniotis' O(M) approach in its method of handling semi-quantitative information. Having been suggested before the above approach, it does, however, not have the latter extended features of O(M) mentioned above.

The **Fuzzy Qualitative Simulation Approach 'FuSim'** by Shen and Leitch [84, 85] adopts the approach taken by QSIM, but, for the purpose of semi-quantitative reasoning, applies fuzzy numbers rather than symbolic landmarks. Using fuzzy sets as representations for both quantities and strength annotations to mathematical relationships between variables allows for the generation of temporal durations.

'Mycroft' (Coghill, Chantler [86]) is a qualitative reasoning framework that allows for different ways of performing a qualitative reasoning task within the same environment. It combines the best features of FuSim with those of other approaches, which are not explicitly considered here, in a single approach (for more details see Coghill, Chantler [86]). The above characteristic of FuSim applies therefore likewise to Mycroft. However, Mycroft requires causality information in addition to the equation set and can either be run semi-constructively or constructively[4] which makes it unique among the compared approaches.

Completely different from the above qualitative reasoning approaches that are rooted in Hayes' Naive Physics concept is the application of *heuristic* **If - Then Rules** (overview by Puppe [45]). Mathematical relations are not necessary for this approach - although they could be integrated. The

[4] The term 'constructive' refers to whether the algorithm uses the system constraints to generate unique output states (i.e., constructive), or to filter out the inapplicable among previously generated states (i.e., non-constructive).

main idea, however, is to collect unstructured knowledge from experience for model building purposes. The general system knowledge is collected in a rule base while facts about any specific situation form the fact base. Rules are evaluated or 'fired' if the facts match their condition - or 'IF' - part. Only one rule is fired at a time and different algorithms exist that determine which rule is fired in case that the condition part of several rules is fulfilled. The result of a rule evaluation is added to the fact base and a new rule evaluation cycle commences by comparing the updated fact base with the condition parts of the rules. The rules are recursively 'chained' in this manner until no more rule conditions are fulfilled. For multiple-phase simulation of such models, time-dependent rules are triggered additionally by a global clock.

Fuzzy Modelling, based on Zadeh's notion [87], covers to a certain extent the functionality of purely rule-based modelling, although the rules are normally not chained. The main characteristic, however, is the facility to handle uncertainty of parameters and decisions in addition. Fuzzy set theory, the basis of fuzzy modelling, is a generalisation of conventional ('crisp') set theory in that it allows for graded set-membership in addition to full and non-membership and by this means deals with uncertainty. Also, the fuzzy approach is aimed at dealing with complex multivariable systems on the basis of relatively limited information.

Likewise independent of Hayes' concept, the **classical system theory** applies qualitative models in the case of insufficient system knowledge. Different general behaviour and stability analyses of systems are possible on the basis of at least structurally defined mathematical relationships. In [56], Lunze introduces simple abstraction operators which are applied to a quantitative state space model to show the close relationship between the abstracted model representations in qualitative reasoning and conventional system theoretical models. Similarly to most of the above approaches, qualitative simulations of continuous processes are carried out using discrete approaches - for example **Automata** [88, 57] or **Petri-nets** (more specifically state machines[5]) [89, 90]. For simulation purposes, however, a mathematical description in the form of abstracted differential equations is normally not necessary. To specify a process model, both approaches require the set of possible system states, possible input settings, an initial condition and an input sequence. Additionally, a transition function must be specified for automata while for Petri-nets, a net topology as well as input conditions for 'labelling' the transitions between the nodes or 'places'[6] must be defined. Both approaches can generate all possible qualitative trajectories, but mixed with spurious solutions. Automata feature a coarse reference to the time scale while the Petri-net approach requires special extensions to represent information about temporal relations. Due to their similarity, state machines and automata can directly be translated into each other.

[5] State machines are a specific class of Petri-nets, in which every transition has exactly one predecessor and one successor.
[6] Each place in the Petri-net represents a semi-quantitative state in terms of ranges.

An interesting suggestion to exploit different kinds of knowledge was made by Kluwe, Krebs, Lunze and Richter [91, 92, 93] who introduced a **Three-Layer Process Model** in which quantitative information is represented in the form of differential equations, while qualitative knowledge is exploited both by a Petri-net and heuristic rules. The ambiguities of the central component, the Petri-net, are resolved in this approach either with the help of the heuristics or by the sets of differential equations that are individually assigned to every place (i.e., qualitative state) in the Petri-net. This approach is particularly useful for large scale systems where sufficient quantitative mathematical information about the process in its various conditions can be acquired.

3.3 Qualitative Models - The Comparison Of Their Characteristics

In this section, the main characteristics of the above introduced qualitative modelling approaches with respect to the application to modelling and simulation in control engineering are discussed. According to their level and kind of abstraction, the approaches are firstly sorted into the three distinct categories, which have been introduced in section 3.1: ***sign-based qualitative causal models***, ***semi-quantitative causal models*** and ***heuristics based semi-quantitative models***[7]. These categories are introduced in order to simplify and shorten the following discussion, since many characteristics are shared among the members of each group.

***Sign-based causal* approaches on the basis of mathematical relations:**

- 'Qualitative Physics based on Confluences' (De Kleer, Brown [58])
- 'Qualitative Process Theory', implemented in 'QPE' (Forbus [59])
- the 'Qualitative Simulation Approach QSIM' (Kuipers [72, 73, 74, 75, 76])
- the 'Qualitative Physics Compiler QPC' (Farquhar [80, 81])

***Semi-quantitative causal* approaches:**

- the 'Qualitative Algebra Q1' (Williams [65])
- the 'Quantity Lattice' (Simmons [66])
- 'Qualitative Process Theory using linguistic Variables' (D'Ambrosio [70, 71])
- 'SIMGEN', an extension to Qualitative Process Theory (Forbus and Falkenhainer [69])
- 'Q2', 'Q3' and 'NSIM' the semi-quantitative extensions of QSIM, (Kuipers and Berleant [77], Berleant/Kuipers [78], Kay/Kuipers [79], respectively)
- the 'Semi-Quantitative Physics Compiler, SQPC' (Farquhar and Brajnik [82])

[7] This is just a simplified classification for the purpose of the following evaluation. The different approaches are classified according to their main concept.

- 'Order-of-Magnitude Reasoning O(M)' (Mavrovouniotis, Stephanopoulos [62] / Mavrovouniotis [63])
- 'Formal Order-of-Magnitude Reasoning, FOG' (Raiman [83])
- the 'Fuzzy Qualitative Simulation Approach FuSim' (Shen, Leitch [84, 85])
- the 'Qualitative Reasoning Framework Mycroft' (Coghill, Chantler [86])
- 'Petri-nets' and 'Automata' (Lunze [90], Bredebusch/Lunze/Richter [89], Lunze [57], Hopcroft/Ullman [88])
- combined quantitative-qualitative modelling using a 'Three-Layer-Model' (Kluwe, Krebs, Lunze, Richter [91, 92, 93])

***Heuristics* based semi-quantitative approaches:**

- If - Then Rules (see overview by Puppe [45])
- Fuzzy Modelling (based on Zadeh's notion [87])

Although the above *heuristics* based approaches are normally not considered as qualitative modelling approaches in the narrow sense, they are included here because they fall into the group of semi-quantitative modelling approaches. In [94], Sugeno and Yasukawa argue similarly and state that "there are small distinctions and big similarities between fuzzy modelling and qualitative reasoning". They stress the advantages of fuzzy modelling in practical applications and suggest a fuzzy based approach to qualitative modelling on the basis of numerical process data.

The difference between the *semi-quantitative causal* approaches which employ the fuzzy theory and the *heuristics*-based fuzzy modelling approach is that the former make only use of 'linguistic variables' as a means of representing uncertainty in the parameters and relationships while the latter additionally applies fuzzy If-Then rules to represent the process models altogether. Rather than applying rules, the *semi-quantitative causal* approaches are still based on mathematical system equations like all other *causal* approaches.

In the following discussion, the members of each of the groups (*sign-based causal, semi-quantitative causal* or *heuristics* based) will often be addressed collectively. For more information on the individual approaches whose characteristics are not explicitly discussed in either of the sections below, please refer to **Table 3-1**, which gives a complete overview with ratings.

This study is structured in aspects **a)** to **g)**, according to the most important characteristics for modelling approaches with respect to the application in control engineering:

a) Ambiguity, Accuracy And Precision Of Predictions
b) The Possible Complexity Of The Modelled Process
c) Temporal Aspects
d) Using The Available Knowledge
e) The Model Formulation
f) Combining Qualitative And Quantitative Models In A Simulation
g) Utilisation Of The Methodologies

a) Ambiguity, Accuracy And Precision Of Predictions

While 'precision' refers in the context of qualitative and semi-quantitative simulation to the tightness of bounds around a predicted behaviour, accuracy is concerned with the generation of all possible behaviours of the partially defined models. Ambiguity normally occurs whenever possible behaviours are 'hidden' between spurious ones.

Some approaches focus on the generation of all qualitatively distinct possibilities and inevitably generate ambiguous results, whereas others aim at the approximation of a single best and therefore precise result, requiring normally some more specific information. It is, hence, arguable that for a more general overview, accuracy and precision should be considered separately in order to fully appreciate the different strengths of the approaches. Because of their close interrelation, however, it was decided to discuss precision and accuracy here in one section: they are 'orthogonal' issues, negatively affecting each other, but not necessarily mutually exclusive.

The *sign-based causal* approaches, which work on the abstraction level of the set {-, 0, +} (i.e., confluences based approaches), allow only for very coarse predictions of possible future states (Figure 3-1). Inevitably, the absence or neglection of any numerical information results in ambiguous predictions of the model's behaviour. This is acceptable in some applications of qualitative reasoning, but it is inappropriate for simulation in the control engineering domain as it is for most other technical applications.

The *semi-quantitative* extensions like Q1 (Williams [65]) and Q2 (Kuipers and Berleant [77]), which integrate some quantitative information in the form of parameter values and measured data, can reduce, albeit not remove, the ambiguity. Their precision is, although improved from results like "somewhere between 0 and $+\infty$" to coarse value ranges, still very low. With D'Ambrosio's linguistic extension to the Qualitative Process Theory [70], both the number of spurious behaviours is

reduced and the precision of the predictions is improved. Although D'Ambrosio's approach appears to be promising, many of the original ambiguities of QP Theory remain, as is detailed in the conclusions of the work.

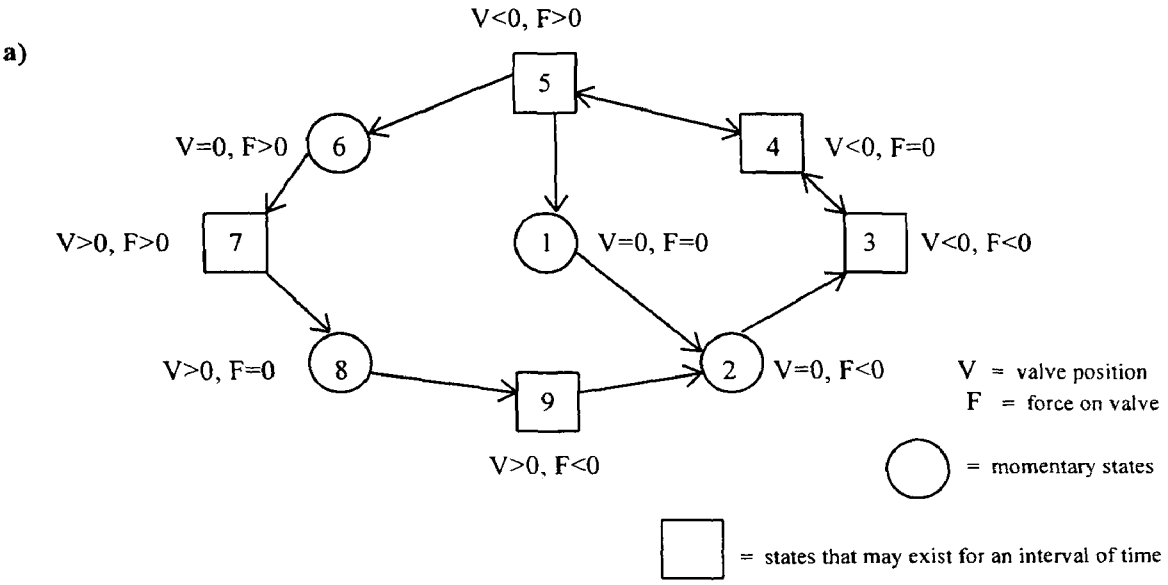
Compared with these early extensions to confluences based techniques, the model-based approaches which are purpose-built for (semi-) quantitative knowledge - particularly the O(M) technique [62], FuSim [84] and Mycroft [86] - are more sound in the different aspects[8] of applying this information. Similarly to D'Ambrosio's approach, FuSim and Mycroft apply a fuzzy quantity space which is particularly suited to the problem domain (i.e., KE for control engineering) since the semi-quantitative knowledge can not only be specified as value ranges but also uncertainty can be expressed through the graded membership of fuzzy sets. At each time increment, the simulation input and output of these two approaches is defined in terms of fuzzy sets. Using fuzzification and defuzzification, the latter approaches could be extended to handle and to yield numerical input and output data, respectively. Although such an extension that trades accuracy for precision would in a way contradict the genuine idea of these approaches and also imply an unrealistic exactness, it could enable a more flexible use of the models which will be discussed under aspect **f)**.

An example for an approach that sacrifices accuracy for precision is SIMGEN, which generates numerical models from qualitative descriptions using a mathematical model library, a semi-quantitative domain library and a matching algorithm.

With respect to ambiguity, accuracy and precision, Q3 and NSIM, the further extensions to QSIM and Q2, have arrived at a level that is so far unparalleled by any other causal approach:
while still retaining the accuracy of the underlying QSIM, the precision of the predictions is here directly related to the precision of the initial conditions[9] . Q3 has also been proven to converge, i.e., with decreasing simulation step sizes, the quality of fit with respect to the actual process behaviour increases under the assumption of precise initial conditions.

The Petri-net and automata approaches have been shown to provide sufficient information to allow for the design of qualitative controllers [57, 89] which are assumed to have only coarse measurement information available. Nevertheless, these process models yield only very general information when applied to simulation, which is due to their remaining ambiguity. In [89], it was shown that using state machines, the proportion between spurious and real solutions cannot be improved by increasing the number of qualitative states. The simulation results are comparable to

[8] For example the propagation of (semi-) quantitative information, the precision of predictions using more aggressive, heuristics based conclusions and the renunciation of abstracting the provided numerical knowledge.
[9] In Artificial Intelligence, this characteristic is known as 'stability'.



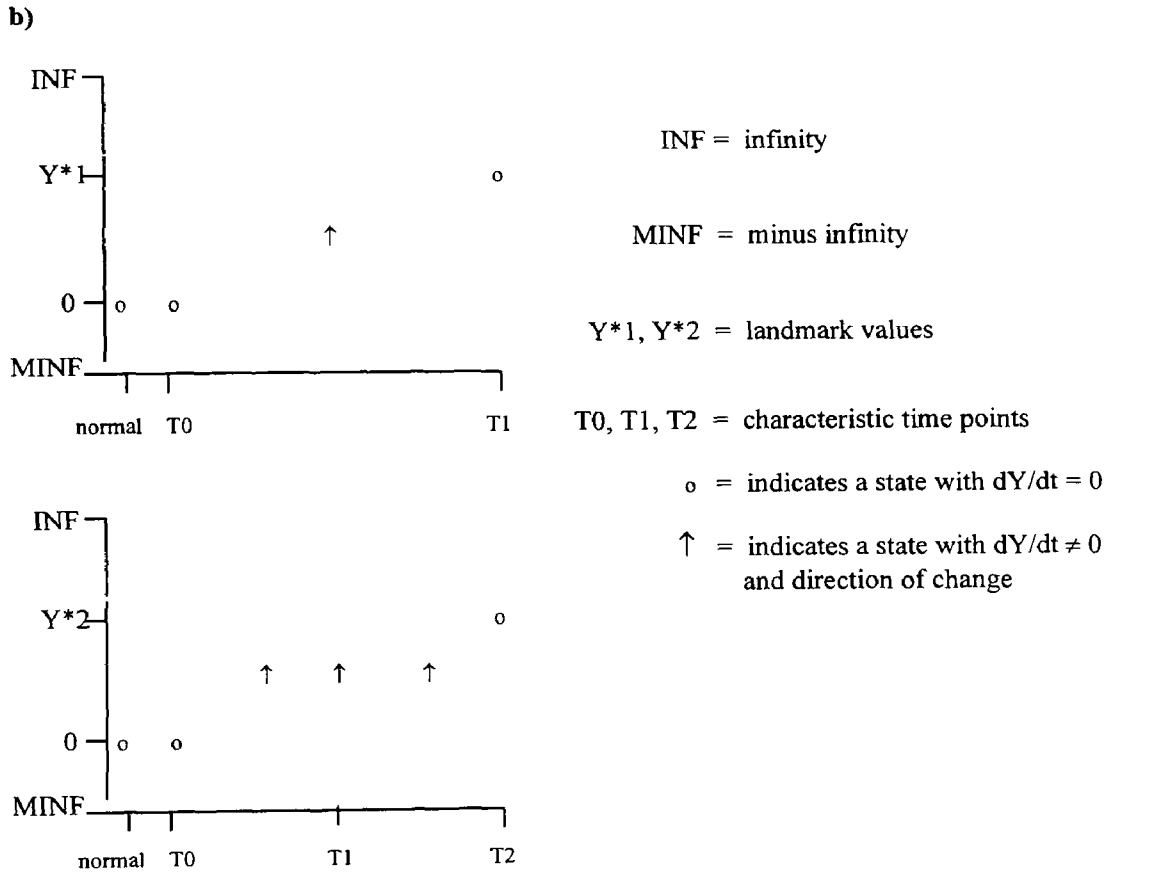


Figure 3-1: Envisioning

a) State Transition Diagram from an example in [58], (©1984 Elsevier Science B.V.)
for a pressure-regulator with continuing input signal, mass, spring and no friction
b)Time-Plot of a Step Response in QSIM [74], (©1988 Elsevier Science Ltd)
possible predictions for a system of two first-order processes in series
Note: Irrespective of the particular processes, the Figure should give an impression of the kind and quality of simulation results!

some of the earlier semi-quantitative approaches in Artificial Intelligence (e.g. Q1, Q2). The combination of Petri-nets with quantitative equations as well as heuristics in the Three-Layer Model is more appropriate for simulation purposes although the sudden switching of system characteristics due to fired transitions in the Petri-net could lead in parts to unrealistic trajectories.

Normally, *heuristics* based models predict unambiguously only a single behaviour in each situation; their accuracy in the above sense is therefore often low while their precision varies significantly, depending on the type of implementation. However, a fine discretisation of explicitly considered states yielding more precise predictions can lead to an exploding rule-base. This disadvantage applies particularly to purely rule-based models and can largely be overcome with the fuzzy approach to *heuristic* modelling. The quality of fuzzy models depends additionally on the definition of the fuzzy membership functions. Within this comparison of qualitative and semi-quantitative modelling approaches, however, the simulation of fuzzy models stands out with respect to precision. The defuzzified, numerical output values can be close approximations of the real process behaviour. These predictions are therefore in a sense false, but close to reality (i.e., precise), whereas for example the confluences-based qualitative models predict the real behaviour (accuracy), but only in very broad terms and hidden between spurious behaviours (ambiguity). Although QSIM with all its extensions (Q2, Q3, NSIM) is advantageous to the fuzzy approach with respect to accuracy, it can only catch up with respect to ambiguity and precision when the conditions are numerically fully defined.

b) The Possible Complexity Of The Modelled Process

This aspect concerns the possible complexity of processes to be modelled using the different approaches, irrespective of the reasons for limitations. Such reasons would have to be taken into account in a general overview. In the specific context of this work, however, the 'possible complexity' is of such paramount importance that the 'good excuses' for not addressing this aspect appropriately are considered irrelevant.

Modelling is closely related to abstraction, since every model is an abstraction of the real system. The mutual implications between system complexity and abstraction levels is briefly summarised before the qualities of the different modelling approaches are discussed. In the context of qualitative and semi-quantitative modelling, two types of abstraction and their implications on the modelling of complex systems must be distinguished:

(I) - The more generally applied meaning of 'abstraction' refers to taking a step back and considering a complex system from a more global point of view, neglecting the details of the individual components that make up the system. Modelling approaches that allow for this type of

abstraction can deal quite easily with complex systems because the model complexity is not directly related to the system complexity. While approaches that build on physical balance equations are unable to abstract in this sense, *heuristics* based approaches are particularly suitable for this type of abstraction.

(II) - The other understanding of 'abstraction' which is mostly used in qualitative modelling refers to coarse signals and process models whose parameters in the equations are not numerically exactly known but only in terms of value ranges, orders of magnitudes, or, in the most 'abstract' case, in terms of their signs.

For increasing system complexity, this second type of abstraction imposes a major limitation - particularly on some of the *causal* approaches. It can generally be stated that the more abstract (II) the modelling level, i.e., the less numerical information is available or used, the more possible qualitative trajectories can be distinguished for a single set of initial conditions. Also, it is inevitable that with an increasingly abstract (II) simulation, the amount of additional, spurious behaviours generated by the simulation environment increases. All these problems increase dramatically with the complexity of the modelled process unless the modelling approach allows for abstraction in the first sense, taking a more global view of the system. Although all *causal* approaches have to fight with the same problems, they differ in the attempts and success in keeping the complexity dependent ambiguity manageable.

Bearing this in mind, it is therefore not surprising that the most abstract approaches that cannot handle numerical information are very limited with respect to possible complexity. Although these confluences-based *causal* approaches (a most notable example being QSIM, which was repeatedly applied to comparably complex situations) cope respectably, they are inappropriate for modelling in control engineering since they cannot make use of any numerical or semi-quantitative information that is normally available in this domain.

Even many of the *semi-quantitative causal* approaches, like QSIM together with the post-processor 'Q2' have their limitations when the number of remaining spurious predictions becomes unmanageable (Dalle Molle [95]).

Among the *semi-quantitative causal* approaches that handle the information more efficiently and largely avoid increasing ambiguity with increasing complexity are the order-of-magnitude approaches, FuSim and Mycroft, as well as QSIM with Q3 and NSIM and the system science approaches automata and Petri-nets. The most suitable approach for complex situations, however,

appears to be the Three-Layer-Model provided that sufficient quantitative mathematical information about the process in its various conditions can be acquired.

The rather low rating of SIMGEN in **Table 3-1** is not related to the generation of spurious behaviours since it performs essentially a numeric simulation, but it is due to the fact that it relies entirely on a mathematical component library which can easily reach its limits - particularly the more complex the systems become.

Using *heuristics* based approaches, the modeller is free to decide on the type (I) abstraction level of the process model, as it was mentioned above. Even very complex processes can therefore be modelled in a simple way without increase of ambiguity if only general predictions for the main parameters are needed. Also, there is no **direct** correlation between process complexity and possibly extensive modelling effort, but mainly between the required precision of the simulation and the modelling effort. In the case, however, that a significant number of discrete states needs to be considered in purely rule-based process modelling in order to yield satisfactory precision, an 'exploding' rule base (see aspect a)) could result. The fuzzy based approach with its nonlinear interpolation facility for continuous output states does not suffer so much from such problems as it requires only a comparably coarse 'grid' of system states to be considered in the rules.

c) Temporal Aspects

The *causal* qualitative multiple-phase simulation techniques are mostly based on a similar notion of time: time is composed of intervals that may be related in different ways, like one interval being before, after or equal to another. Likewise, 'histories' are normally applied to represent how things change through time. A particularly important characteristic is the projection of 'trends' for the purpose of coarse temporal predictions.

Despite this common ground among the model-based (*causal*) approaches, there are several differences. Particularly the early works, for example by Forbus, De Kleer and Brown, are limited in their ability to reason about time, as Williams [96] points out. He developed therefore his *semi-quantitative* 'temporal constraint propagator' (TCP), which is based on the notion of 'value history' and was implemented in Simmon's 'Quantity Lattice'. The TCP improves the maintenance of a consistent partial order of time points by assigning durations to the qualitative states and answers queries about relationships between time points. The consideration of time delays and temporal reasoning for feedback systems are among Williams' particular contributions.

One of the more advanced approaches among the *sign-based causal* methodologies is 'QSIM' in that it uses a 'standard' mathematical model of time [75]. Also, QSIM differs from the other

approaches in allowing new landmarks to be discovered during the qualitative simulation, thus creating additional time points[10] as new qualitative distinctions on the time scale. This yields better information on characteristic states such as increasing, decreasing or stable oscillation. Nevertheless, the lack of a representation for time delays is a deficit of QSIM. Since QPC actually employs QSIM, it shares most of these characteristics.

Despite their advances, neither TCP nor QSIM allow for truly dynamic considerations because of their inability to determine the time needed for state transitions.

Among the *semi-quantitative causal* approaches, neither the order-of-magnitude approaches take temporal aspects into account since they perform only algebraic reasoning, nor does D'Ambrosio's fuzzy logic extension to qualitative process theory provide any information on temporal durations. In fact, only few of the approaches feature good representations of dynamic behaviour. Both FuSim (Figure 3-2) and Mycroft take account of the time that an output of interest remains in a distinguished state as well as of the time for its transition to the next distinguished state (i.e., persistence- and arrival time, respectively). Also, the iterative refinement of time steps in Q3 as well as the three approaches, which apply in parts numerical simulators, i.e., NSIM (Figure 3-2 c), SIMGEN and the Three-Layer-Model, yield good results.

With respect to the simulation of *heuristics* based models, temporal aspects are differently included in purely rule-based and fuzzy approaches. Rule-based multiple-phase simulation applies a global clock that advances the simulation process in discrete, numerical intervals of time and initiates state changes by the activation of time-dependent rules. The length of the intervals is application dependent and may be constant or variable. For the resulting new parameters of the system, the time-independent rules are evaluated afterwards and further parameter values are calculated. The clock is advanced and this sequence is iterated as long as time-dependent rules are triggered. With this concept of rule-based multiple-phase simulation, a determination of time required for state changes is therefore possible.

Although this has not yet been tried, the above described discrete multiple-phase simulation approach could likewise be applied to fuzzy models to handle additionally uncertainty with respect to parameter values. The most common technique to yield a dynamic fuzzy model is to apply the derivatives or integrals of input or output variables as additional inputs to the fuzzy system. Using first order derivatives as additional inputs, however, results normally only in a very coarse, 'dynamic-like' behaviour (Figure 3-2), while higher order derivatives become unmanageable for

[10] Distinguished time points are those points where something important happens to the value of the function, such as passing a landmark value or reaching an extreme [75].

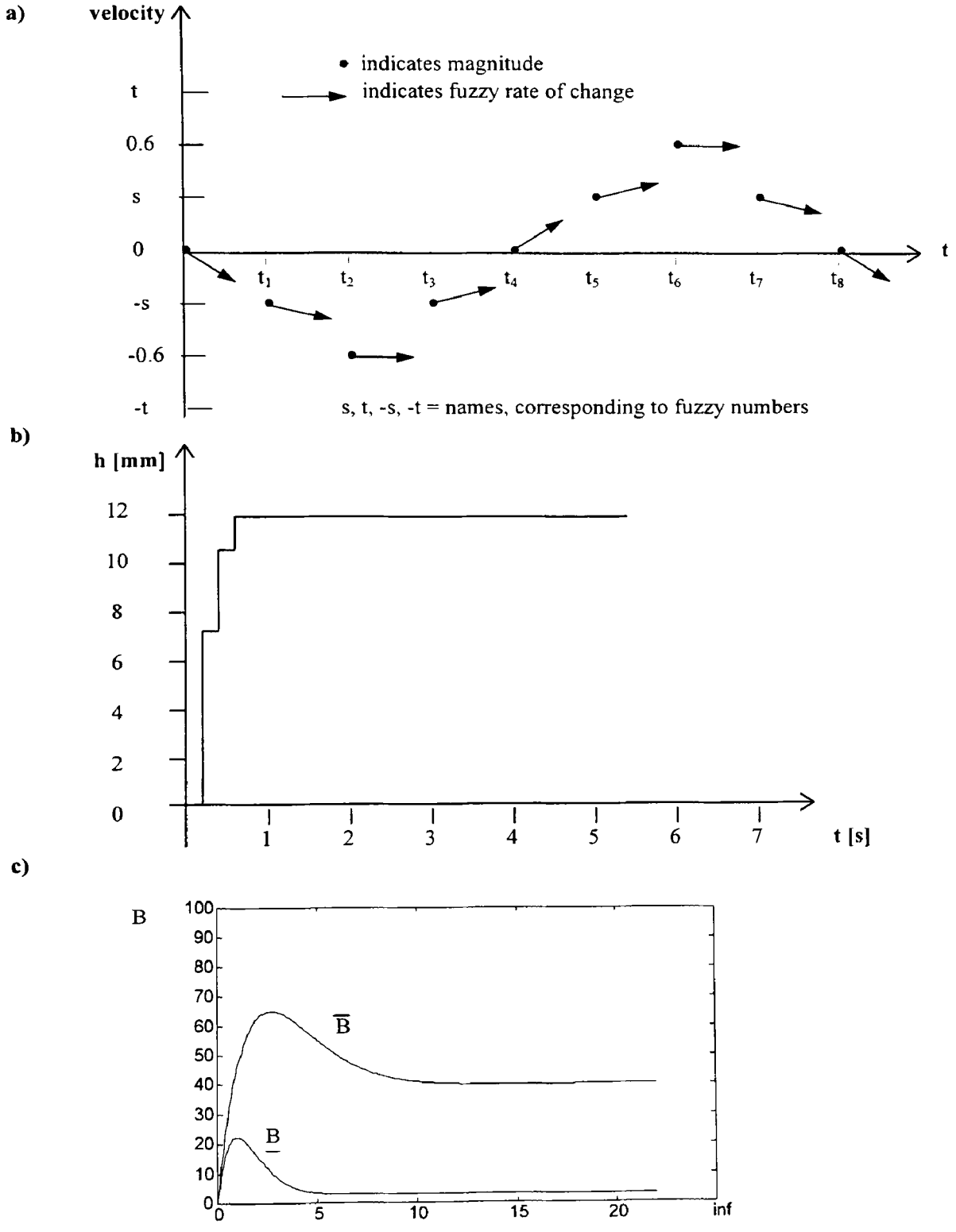


Figure 3-2: Results Of Semi-Quantitative Simulation

a) Time plot in FuSim [84], ©1993 IEEE: velocity of a 'mass on a spring' system
b) The step response of a fuzzy model: the output derivative is fed back as additional input
c) NSIM output [79], ©1993 AAAI: Dynamic envelopes defining the lower bound $\underline{B}(t)$ and the upper bound $\overline{B}(t)$ on $B(t)$

'manual' modelling (i.e., formulating the rule base and defining membership functions). Only process data based identification approaches can use the potential of higher order derivatives. Nevertheless, other approaches have been applied to consider dynamic aspects in the normally static fuzzy concept. Still though, these concepts are either not truly dynamic approaches[11] or are not generally applicable[12] as shall be seen in the following chapter.

d) Using The Available Knowledge

De Kleer and Brown stated that by taking the qualitative approach, an often significant amount of knowledge loss cannot be avoided [58]. In contrast to this understanding, the following statement by Mavrovouniotis [62] describes best the idea of applying any of these modelling techniques in control engineering:

> "One of the motives for using AI-methods is the desire to apply as much of the available knowledge as possible, despite the disparity in the forms of the knowledge involved."
>
> Mavrovouniotis [62]

The keywords in this sentence are "as much as possible" and "available".

In this comparison of modelling approaches, the focus is therefore put on both the flexibility of the approaches to incorporate different kinds of knowledge and their particular minimum requirements in the light of the likelihood of their availability in a practical modelling context. In the domain of process modelling for control engineering in industry it is assumed here that at least some sort of semi-quantitative information is readily available, while the mathematical system equations are not always at hand.

By definition, *heuristics* based models enable the usage of experience gained with the real process. Also, they allow for the integration of other different types of knowledge - even mathematical relations and, most notably, quantitative and semi-quantitative knowledge which is usually available in the engineering domain. Although *heuristics* based models are not very efficient in exploiting *causal* information, they fulfil the aspect of using the available knowledge overall very well.

The *sign-based causal* approaches which neglect any (semi-) quantitative knowledge clearly do not comply with Mavrovouniotis' above cited maxim.

[11] For example the programming of standard trajectory patterns [97, 98].
[12] For example averaging locally valid trajectories of linear dynamics to yield a global system behaviour (Sugeno, Kang [99]) is often erroneous as soon as oscillations occur because of cancellation effects due to phase differences.

In general, the *causal* approaches are only applicable if the main physically meaningful relations are known - additionally, some of the approaches (like QSIM and Mycroft) require explicit causality information. It is this characteristic that implies the major disadvantages of 'deep' knowledge[13] representations. With respect to the *causal* modelling approaches, it is interesting to note that hardly any attempts were made to overcome these disadvantages, because the acquisition of the 'deep' knowledge was explicitly seen as being outside their scope.

Among the *semi-quantitative causal* approaches, Mavrovouniotis' O(M) approach is a step towards more flexibility as it allows for integrating some *heuristic* relationships in addition to mathematical ones. This advantage, however, comes at the expense of loosing the strict causality. Another approach that combines different forms of knowledge is the Three-Layer- Model, although its disadvantage is the requirement of a significant amount of quantitative mathematical information.

e) The Model Formulation

Heuristics based knowledge is normally available and intuitively understandable as the rules are usually formulated in plain English. However, even the application of the most recent graphically oriented fuzzy shells requires the theoretical understanding of the modelling methodology. In fact, fuzzy modelling can be highly complex with a wealth of parameters to set and algorithms to choose from. Without guidance, the inexperienced modeller will find it very difficult to translate his/her unstructured, practical experience into a structured rule base. The main problem with respect to purely rule-based modelling is closely related to what was discussed under aspects **a)** and **b)**: the need for often highly complex rule-bases makes the model formulation very cumbersome and error-prone.

The *model-based (causal)* approaches often necessitate a very thorough theoretical understanding of their underlying concept and a specific descriptive language or input format must often be learned. To maintain causality, the formulation of many 'trivial' constraints is in some cases obligatory which makes the formulation complex and cumbersome [61]. The PhD-thesis by Dalle Molle [95] on the investigation of the modelling and simulation of chemical processes using QSIM concluded that qualitative model building was not a well-defined procedure but something of an art. These problems are not uniquely related to QSIM but apply likewise to the other *causal* approaches as Rajagopalan [100] indicated. More recent research at the University of Texas (e.g. Farquhar, [80]) aimed therefore at the simplification and partial automation of the model building procedure but resulted only in quite limited improvements. Farquhar's simplifications are based on the application of fragment libraries for different domains. Experience from botany and chemical engineering which

[13] 'Deep' knowledge refers to causal knowledge, as opposed to 'shallow' (i.e., heuristic) knowledge which does normally not represent the causal nature of an observation.

was gained by other researchers applying QPC showed, however, that "libraries of realistic model fragments ... are extremely difficult to construct" [80]. The library based approach, which is not related uniquely to any particular modelling approach[14], is certainly one of the most effective means of simplifying modelling. Nevertheless, it is important to consider that any library is limited and simplified extensions to such libraries should be possible. Since hardly any libraries for industrial process modelling are developed so far, the rating in **Table 3-1** considers both the use of a library (if applicable) and the ease of its extension.

In Mycroft, a macro function is applied which translates a textual model specification into the correct format. Although this greatly simplifies the model definition it still requires some familiarity with the required textual format.

f) Combining Qualitative And Quantitative Models In A Simulation

Ideally, it should be possible to compose a structured process model arbitrarily from components of different model types. The direct implication of combining a qualitative process model for example with a conventional transfer function for simulation purposes is the need for precise numerical inputs and outputs of the individual components.

According to the discussion in **a)**, fuzzy models with continuous numerical input and output variable ranges fulfil this requirement as well as the Three-Layer-Model and SIMGEN, while Q3 and NSIM yield normally less precise results in favour of accuracy. If used within a general purpose simulation environment, however, both the fuzzy and the Three-Layer-Model approach are easier to interface with conventional models than SIMGEN or Q3/NSIM.

It would also be possible to extend the order-of-magnitude approaches and particularly the fuzzy quantity space based approaches FuSim and Mycroft so that numerical outputs are obtained, while the use of numerical input values is anyway unproblematic. Although the linguistically extended Qualitative Process Theory [70] is also based on a fuzzy quantity space, the defuzzification of its outputs in order to obtain numerical values would not be sensible due to the ambiguities of this approach.

It must be stressed here that precise numerical output values of any qualitative approach always bare the risk of being misinterpreted as exactness, although they are actually approximations whose quality depends on the amount of knowledge provided as well as the algorithm used.

[14] Among the qualitative modelling approaches, basic libraries have been developed for example by De Kleer/Brown and Forbus.

The only way to combine the remaining *causal* qualitative models with model components of other types would be in series, with the *causal* qualitative model always at the end of the structure.

g) Utilisation Of The Methodologies

This aspect is primarily concerned with the availability of program modules, code or implementation recipes for the different qualitative modelling and simulation methodologies. As a secondary factor, the ease of implementation is also considered.

Heuristics based approaches are featured in many commercially available programs like expert systems or fuzzy shells. Nevertheless, only few purpose built frame-based expert systems and fuzzy process modelling tools for control domain relevant simulation of industrial processes exist[15]. Among the more recent programs which allow for the specification of fuzzy models as well as their simulation in arbitrary combinations with conventional process models, Matrix$_x$™ and MATLAB™ (with fuzzy toolbox) should be mentioned. Another advantage of the fuzzy modelling approach is its detailed documentation in the literature as well as its relative simplicity which allows for quite quick and simple individual implementations of the whole modelling and simulation approach, if necessary.

Among the system science based *semi-quantitative causal* approaches, only Petri-net modelling and simulation programs are commercially available. Individual implementations of Petri-nets - possibly within general purpose simulation environments - are laborious but possible due to the widespread expertise and documentation. Although also based on well established approaches, the Three-Layer-Model is more difficult to implement individually as the interplay between the components is of key importance. It is therefore more sensible to build on the extensive work on the implementation of "DEMSI", a simulator for the Three-Layer-Model, which has mainly been carried out at the Technical University of Hamburg-Harburg[16]. The development of "DEMSI" has particularly focused on its online application in advisory systems for complex plants.

The situation is different for the artificial intelligence-rooted *causal* approaches as these are not based on a standard concept. In order to make use of them it is necessary to fall back on the existing programs since they consist of comprehensive sets of reasoning algorithms which resulted from individual research projects. Since most of the existing programs are normally only research implementations which were used for validation purposes, they have often never been in a very usable form or the code has been abandoned after the research projects have been finished. This

[15] For an overview see Linkens [101] and Puppe [45].
[16] For further information on DEMSI please contact Prof. Dr. Jan Lunze or Dr. Henrik Richter, Technical University Hamburg.

aspect reduces dramatically the group of potentially applicable approaches to QPC, SQPC, Mycroft and QSIM, including its extensions Q2, Q3 and NSIM.

The QSIM program has been maintained and updated and is available in LISP source code via 'ftp' from the file server at the University of Texas at Austin[17], together with its extensions. A good deal of experience in compiling and installing such systems is, however, required. The same applies to QPC and SQPC[18].

Mycroft is currently being finalised in Common LISP code and will be available via 'ftp' from Heriot Watt University in Edinburgh[19]. Similarly to QSIM, the installation will probably require some experience. A re-implementation of Mycroft in C++ which is also being considered at Heriot Watt University could simplify the installation and improve both the portability and speed and therefore further increase the attractiveness of this interesting approach for a broader community.

3.4 Summary And Selection

For a quick reference, the results of the discussion in the preceding section are summarised in the following **Table 3-1**. The ratings which range from 'very poor' (- -) to 'very good' (++) should give a general feel for the comparison between the approaches with respect to the considered characteristics a) - g). From this table, it is quite obvious that *sign-based causal* approaches are unsuitable for practical control engineering purposes which is generally due to their over-abstraction of readily available (semi-) quantitative information.

Despite its deficiencies with respect to dynamics, the fuzzy based modelling approach turned out to be advantageous in almost all aspects. However, since the representation of process dynamics is generally not a strength of the qualitative and semi-quantitative modelling approaches, fuzzy modelling is not really an exception as it was discussed above. Overall, this approach is therefore the first choice.

Among the *semi-quantitative causal* approaches, the Three-Layer-Model, Mycroft and QSIM with Q3 and NSIM are the most appropriate approaches for application in control engineering which have not only advantages and disadvantages with respect to each other, but in particular with respect to the fuzzy approach. Depending on the individual emphases, any of these three *causal* approaches could be a sensible complement to the fuzzy modelling approach.

[17] Please contact Prof. Dr. Benjamin Kuipers, University of Texas at Austin, for information on the terms of usage.
[18] Please contact Prof. Dr. Adam Farquhar, Stanford University.
[19] Please contact Dr. George M. Coghill, Heriot-Watt University / Edinburgh, for further information.

	sign-based causal			semi-quantitative causal												heuristics-based	
	QPhys	Qproc (QPE)	QPC	QSIM incl. Q2	QSIM, Q2,Q3, NSIM	SQPC	Qlattice	Q1	SIMGEN	Q(M)	FOG	QProc + fuzzy	FuSim	Mycroft	PetriNet, automat a	Three-Level Model	rule based	Fuzzy
a) precision, accuracy ambiguity	-	-	-	o	++	o	o	o	+	+	+	o	+	+	o	+	+	++
b) possible complexity	-	-	-	o	+	o	+	o	o	+	+	o	+	+	+	++	o	+
c) temporal aspects	-	-	o	o	+	o	o	o	+	- -	- -	-	+	+	o	+	o	o
d) using available knowledge	- -	- -	-	-	-	-	-	-	-	+	-	-	-	-	-	o	+	++
e) model formulation	-	-	-	-	-	-	-	-	-	-	-	-	o	o	o	-	-	o
f) combination with quantitative models	- -	- -	- -	- -	o	- -	- -	- -	+	-	-	- -	-	-	- -	++	o	++
g) utilisation	- -	- -	o	o	o	o	- -	- -	- -	- -	- -	- -	- -	o	+	o	+	++

Table 3-1: Evaluation Of Qualitative Modelling And Simulation Approaches

3.5 Conclusion Of The Survey

In this survey, the ideas behind qualitative modelling which is being pursued by several research communities have been clarified. Different abstraction levels of qualitative models as well as the most commonly used knowledge types - *causal* and *heuristic* knowledge - have been illustrated in the introductory section in order to specify the three categories into which the considered qualitative modelling approaches have subsequently been split; *sign-based causal* approaches, *semi-quantitative causal* approaches and *heuristics based semi-quantitative* approaches. These three groups of approaches cover the whole range of existing qualitative modelling approaches.

Although some of the main techniques within each class have been introduced and compared, this overview does not claim to be complete with respect to individual approaches. Also, it should again be stressed that all comparisons and ratings are made with the application to modelling and simulation in control engineering in mind, with a particular emphasis on knowledge handling. Therefore, it is well possible that approaches which collected mainly bad ratings (**Table 3-1**) have characteristics that are exceptionally well suited to other tasks. SIMGEN, for example, has been particularly designed for self-explanatory simulations: any question about the system behaviour of the type "What if ...?" or "Why ...?" can be asked and is automatically answered by the program, which makes it an interesting approach for tutorial purposes.

Some of the earlier works, like the 'Qualitative Physics based on Confluences' or the 'Qualitative Process Theory' are known to be quite limited in their abilities [102]. Nevertheless, they had to be considered in this overview because they have largely influenced the other approaches and are therefore essential for the understanding of the still ongoing research.

This cross-section of relevant qualitative modelling techniques showed that among all *causal* approaches, only the artificial intelligence approaches Mycroft and QSIM with its latest extensions, as well as the largely overlooked system science-rooted Three-Layer-Model are applicable for modelling and simulation in control engineering. Overall, however, fuzzy modelling got the best ratings in the different categories of this comparison. Based on Zadeh's early papers from the late 1960's, fuzzy modelling is not only the most promising but also one of the oldest approaches dealing with qualitativeness. Still though, fuzzy modelling is an active field of research.

On the basis of this analysis of qualitative modelling approaches and their shortcomings, it was decided to focus on fuzzy modelling as a means of handling partial knowledge of processes and their behaviour within the integrated modelling approach. In order to improve the fuzzy-based

modelling of *dynamic* processes, a hybrid modelling concept was developed, which is introduced in the following Chapter 4. To simplify the formulation of this fuzzy hybrid model, some important standards and defaults are specified, too. Finally, with the integration of the fuzzy hybrid modelling approach into the overall Knowledge Engineering methodology (Chapter 5) and the implementation of the latter in an interactive software environment (Chapter 6), the formulation of *dynamic* fuzzy models was fully automated. Thus, the two shortcomings of 'fuzzy':

c) temporal aspects and

e) model formulation,

which have been discussed in this chapter, are systematically addressed.

Despite the advantages of the fuzzy-based approach, it must not be considered as the 'one and only' solution but it should eventually be complemented by one of the prime *causal* approaches for situations where purely *causal* qualitative modelling and simulation is required. QSIM with extensions, Mycroft and the Three-Layer-Model are the most promising candidates for the future extension of this work.

4. The Fuzzy Hybrid Modelling Approach For Nonlinear Dynamics

In this chapter, a new, simplified approach to the modelling of multivariable nonlinear dynamic processes on the basis of restricted process knowledge is introduced[1]. This approach is aimed at overcoming the limitations of fuzzy-based modelling of *dynamic* processes which have been discussed in the previous chapter. After pointing out the merit of applying a fuzzy hybrid concept for this purpose, an overview of the previous work in this particular area is given. In the stepwise modelling sequence, which is illustrated with an example, some default settings are introduced. These defaults play an important role in the simplification and standardisation of the suggested modelling approach. Using examples, the advantages of the new approach in comparison with the successful approach by Takagi and Sugeno [103] as well as its applicability to real processes is demonstrated.

For a general introduction to the fuzzy theory, please refer to [104] and for more details to [105] or the pioneer work by Zadeh [87, 106].

4.1 Preliminary Considerations

The association between input and output variables in terms of plain fuzzy rules of the kind "IF *A* is *small* AND *B* is *big* THEN *C* is *medium*" is a purely static relationship; a fuzzy controller - or model - can therefore be fully represented by a static characteristic in form of one or more surface plots that relate the output directly to the inputs. Such surface plots are a standard viewing option in virtually all fuzzy modelling tools. Nevertheless, it is possible to model continuous dynamic processes using the basic fuzzy modelling approach with higher order derivatives (discrete: z^{-n}, $n \geq 1$) as additional inputs. This kind of dynamic modelling has, however, its drawbacks:

- The required rules become quite complex and cannot be formulated on the basis of experience anymore - only the identification of the fuzzy model using measured data is feasible.
- Analyses (e.g. stability) are hardly possible.
- Very small discrete simulation increments are necessary in order to achieve satisfactory continuous effects.
- The simulation step size influences the response of the model.
- In industry, the acceptance of purely fuzzy-based systems is low.

[1] This proposal of a new Fuzzy Hybrid approach has been submitted and accepted for publication in the IEEE Trans. on Systems, Man, and Cybernetics. The revised manuscript is largely identical to this chapter and is appended to this thesis for reference.

Another limitation of the basic fuzzy modelling approach is the saturation effect of the output values at the borders of the specified operating range. The output *C*, for example, is not extrapolated beyond the numerical values associated with the fuzzy sets "MAX" or "MIN".

However, the above limitations and drawbacks which are of importance for the envisaged application can be overcome by combining the fuzzy notion with linear dynamic system equations to a fuzzy hybrid system. An important advantage of such combined systems is the possibility of making use of the various traditional system analysis techniques.

The locally valid, piecewise linear SISO system equations can either be obtained through very basic local process identification experiments or derived from the practitioner's experience, using an 'intelligent' interface (cf. [107, 108] and the following chapters of this thesis). This particular kind of 'patchy' information that is required for the approach introduced in this chapter is a unique feature of the proposal as is its simplicity. Furthermore, the proposed approach is particularly aimed at continuous processes whose dynamic characteristics vary, depending on one or more parameters, which is frequently the case in process industry. This type of truly 'nonlinear dynamic modelling' addresses therefore significantly more complex situations than the commonplace understanding of 'nonlinear (and) dynamic modelling', where the dynamic behaviour as such is linear and only preceded (or succeeded) by a nonlinear static characteristic (e.g., the simplified Hammerstein and Wiener-models, respectively [109, 110]).

4.2 An Overview Of Fuzzy Hybrid Approaches

The idea, to make best use of both fuzzy and conventional mathematical approaches, is not new. In the remainder of this section, existing fuzzy hybrid approaches are briefly summarised:

In order to analyse the effects of uncertainty with respect to the parameters of ordinary linear mathematical transfer functions or state space models, the fuzzy notion is applied for example by Jain [111], Grobbelaar [112] and Kandel [113], Rouhani/Tse [114], respectively. The application of fuzzy sets as an uncertainty measure necessitates the definition of fuzzy operators analogous to arithmetic operators on rational numbers because the uncertainties have to be carried through the numeric simulation and the effects of several uncertain transfer function parameters must be combined, increasing the overall uncertainty in the simulation result. This approach is mainly based on replacing the ordinary, crisp parameters by fuzzy numbers according to the fuzzy extension principle [115], and therefore it does not require a rule-base. The uncertainty in the simulation result is represented by a set of system trajectories with different probability annotations. The response to

a crisp input is therefore a fuzzy signal (Figure 4-1). Although this is an interesting and important application of fuzzy-hybrid systems, it does not take advantage of the nonlinear modelling capabilities of the fuzzy approach.

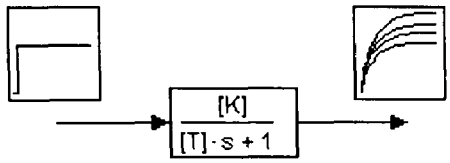


Figure 4-1: Fuzzy Hybrid Model To Express Uncertainty In The Parameters
The Square Brackets Indicate Here Fuzzy Parameters.

Aiming at modelling highly nonlinear processes in a simplified manner using piecewise linear mathematical equations which are combined via the fuzzy notion (Figure 4-2), the fuzzy hybrid approach suggested by Takagi and Sugeno [103] has a different motivation, which is very similar to the aim of this work.

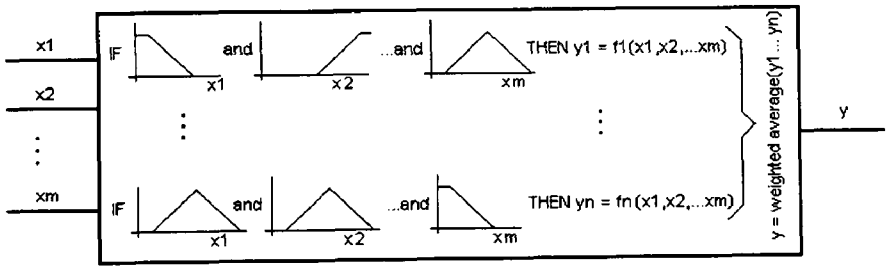


Figure 4-2: The Takagi-Sugeno Fuzzy Hybrid Model

Both input and output of these process models, which are aimed at the model-based design of multivariable fuzzy control systems, are non-fuzzy - or 'crisp'. The particular characteristic of this approach is the format of the fuzzy implication where the "THEN"- part of the rules does not assign a specific fuzzy set from the output space to the output variable as is normally the case but defines the output variable in terms of a function of the input variables. The general format of a fuzzy multiple input - single output process law (analogous to a fuzzy control rule but **applied to modelling**) is defined as follows:

$$L^i: \quad \text{IF } x_1 \text{ is } A^i_1,\ x_2 \text{ is } A^i_2,\ \ldots\ x_m \text{ is } A^i_m,\ \text{THEN } y^i = c^i_0 + c^i_1 x_1 + c^i_2 x_2 + \ldots + c^i_m x_m$$

with L^i denoting the i-th process law, c^i_k coefficients, A^i_k fuzzy sets, x_k input variables and y^i the output from the i-th process law. Using the truth value w^i of the premise of the i-th process law, which is calculated as

$$w^i = \prod_{k=1}^{m} A^i_k(x0_k) \quad ,$$

a given input $x0 = (x0_1, x0_2, \ldots, x0_m)$ yields the overall output

$$y0 = \frac{\sum_i w^i y^i}{\sum_i w^i} \quad .$$

The overall output of the fuzzy model is therefore the weighted average of the y^i's.

This approach, which allows for highly nonlinear modelling despite the small number of rules needed, is widely applied and acknowledged, with the MATLAB™ fuzzy toolbox probably being one of its latest implementations. The modelling approach was repeatedly shown to be advantageous in conjunction with fuzzy identification [103, 99] and applied, for example, to helicopter flight control [97].

Compared with other rule-based approaches, a particular strength of the Takagi-Sugeno approach is that normally only very few antecedents among all possible input-combinations are required, which reduces the size of the rule base significantly.

The Takagi-Sugeno model has, however, an important limitation: combining piecewise dynamic system equations, for example transfer functions, differential or difference equations in a similar fashion as the linear static equations above is not generally possible. Although Sugeno and Kang [99] have actually applied this modelling approach for the combination of first and second order difference equations, which are valid under certain process conditions, this is not **generally** permissible. Using dynamic system equations in the consequent, or 'THEN', part of the Takagi-Sugeno model can lead to erroneous results whenever oscillating signals occur, either through system equations with complex poles or due to a frequency input signal. Since the overall output of the system is determined by averaging trajectories, phase differences lead, for example, to cancellation effects and therefore spurious predictions (cf. section 4.3.2).

Kuipers and Aström [116] describe a heterogeneous control system that switches between different local control laws[2] using the fuzzy notion to achieve smooth transitions between adjacent regions. The global heterogeneous control law in this approach is defined as the weighted average of the local control laws, where the weights are returned by the fuzzy membership functions. This concept is therefore very similar to the Takagi-Sugeno model, the main difference being the separation between the fuzzy part, which is responsible for the soft transitions by determining weighting factors, and the actual control laws as well as the aggregation of the separate local outputs (Figure 4-3). Due to the summing of the local outputs, this approach is likewise limited to static equations for the local control laws (see section 4.3.2). The Takagi-Sugeno model is, however, more compact and computationally efficient than the approach suggested by Kuipers and Aström.

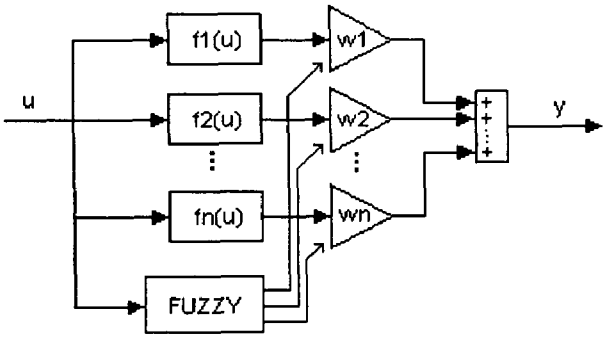


Figure 4-3: The Heterogeneous Control System Approach

Another widely applied approach that combines the advantages of both conventional control engineering and fuzzy techniques is the adaptation of PID controllers via fuzzy adapters [117] (see Figure 4-4).

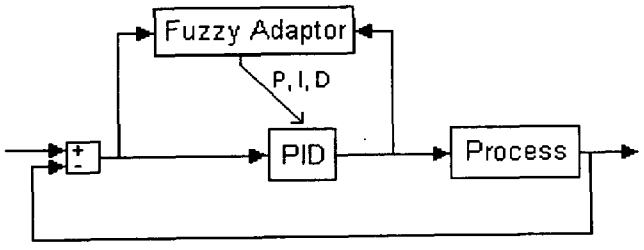


Figure 4-4: Fuzzy Adapter For PID Controller

Like the 'heterogeneous control law' approach, this concept has specifically been applied to the design and implementation of fuzzy hybrid control systems. The success of this approach in many industrial applications [104] is largely based on the wealth of conventional analysis and validation

[2] Kuipers/Aström use the notion 'control law' in the sense of equations rather than IF - THEN rules!

techniques which are important to guarantee the stability of critical processes. This close relation to traditional techniques also increases the acceptance of such hybrid systems in industry.

4.3 Suggestion Of A New Fuzzy Hybrid Approach To Process Modelling

The proposal is closely related to the Takagi-Sugeno model in that it is likewise aimed at expressing very nonlinear functional relations in a simplified, efficient manner using piecewise linear equations that are combined via the fuzzy notion. Both input and output of this model are therefore normally also 'crisp', although an extension to express parameter uncertainty similarly to Grobbelaar [112] would be possible.

To make the fuzzy-hybrid approach applicable for dynamic system modelling and simulation, the major aim of the proposed modelling concept is to overcome the above mentioned limitations with respect to the integration of any kind of dynamic system equations. In spite of the close relation to the Takagi-Sugeno model in terms of its aim and functionality, this process modelling approach is actually derived from the fuzzy adaptation of PID controllers as Figure 4-5 shows.

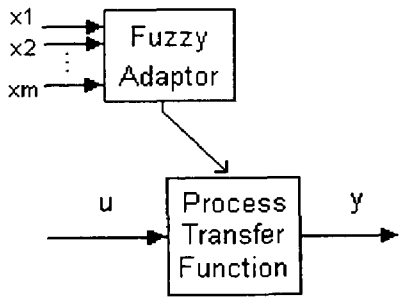


Figure 4-5: The Suggested Fuzzy Hybrid Approach To Process Modelling

Instead of varying the P, I and D parameters of a controller using a fuzzy adapter, the application as a compound simulation model of the process itself requires the consideration of any polynomial or physically meaningful parameter, like time constants and damping ratios. As will be shown in the following sections, the adaptation of these latter parameters allows for the simulation of truly nonlinear dynamic processes which is not the case with any of the approaches discussed in the previous section. Rather than calculating local system outputs and averaging these as in the Takagi-Sugeno or heterogeneous control law approaches, the average parameters are first aggregated according to the current operating condition and transferred to the single overall system

equation from which the output of the model is determined. This system equation may be of any kind - especially dynamic (for example a transfer function or differential equation). The proposed approach is therefore called '**fuzTF**', for **fuz**zy Transfer Function.

4.3.1 The Modelling Sequence Using The Suggested Fuzzy-Hybrid Approach

This section details the above stated general idea by introducing a set of modelling steps together with some important selections and defaults that make the approach very efficient and enable its automation while retaining its flexibility. Using these defaults, the user of such an automated approach would not be required to understand the details of fuzzy modelling. For the introduction of the stepwise procedure, a hypothetical example process without a particular physical manifestation was chosen, which is both simple and illustrative, showing the particular characteristics of the approach.

The parameters of the example process vary, depending on the influence parameter 'INFLUENCE'. In addition, the structure changes from 2^{nd} order proportional behaviour at low levels of 'INFLUENCE' to 1^{st} order at high levels of 'INFLUENCE'.

The modelling sequence - steps 1) to 8):

1) *Determination of the influences on the nonlinear behaviour of the process.* These influences which are responsible for the transitions between different characteristics (i.e., system equations) of the process are the inputs to the fuzzy adapter. The preselection of input candidates and the determination of input variables form the 'structure identification I' according to Sugeno and Yasukawa [94]. In the case of multiple influences, only the one that is expected to be most important should initially be considered. If the model quality proves to be insufficient, further influences can be added to the model in a stepwise fashion so that the previous version of the model can always be re-used.

   example: The parameter 'INFLUENCE' of the example process causes the changes in the process characteristic. With 'INFLUENCE', the fuzzy adapter will therefore have one input.

2) *Partition of the input space.* The question

   "How many characteristic behaviours of the overall system can or should be distinguished and which are the related input conditions to the fuzzy adapter?"

must be answered. To keep the modelling effort at a minimum, this number should initially be quite low (typically two to five). If required, the model quality can be increased at a later stage by considering further operating conditions.

example: Three characteristic levels of 'INFLUENCE' (5, 11 and 20) are distinguished.

3) *Collection of the local process equations.* The source of this information could either be any conventional process identification using a small perturbation approach or even a simple identification of the local process behaviour on the basis of experience or simple step response tests together with look-up tables (e.g. Strejc's method [51]). Generally, however, the local equations should be kept as simple as possible.

example:

The following three typical transfer functions are assumed to have been found through a small perturbation approach:

OP1: at INFLUENCE = *5* $$G_5(s) = \frac{10}{9s^2 + 1.2s + 1}$$

OP2: at INFLUENCE = *11* $$G_{11}(s) = \frac{6}{25s^2 + 4s + 1}$$

OP3: at INFLUENCE = *20* $$G_{20}(s) = \frac{15}{7s + 1}$$

("OP" stands for operating point of the nonlinear influence parameter, here named 'INFLUENCE')

4) *The different local process equations with fixed parameters are combined to a single system equation with variable parameters.* If the local process equations are of different order, the highest order equation is chosen.

example: $G_{example}(s) = y / u = b_0 / (a_2 s^2 + a_1 s + 1)$

In applications where the global **absolute** values are required, the computation of the static gain characteristic in a separate equation becomes necessary (cf. section 4.3.3). An example is given in section 4.4.

5) *The membership functions for the input variables to the fuzzy adapter are generated.* For reasons of simplicity, the default triangular membership functions, which are easily generated around the known operating points of the input variables, should be used. Also, the default functions fulfil the constraint $\Sigma\mu_i = 1$ for any input in the operating range. With these simplifications, the fuzzified inputs contain already the complete information for the weighted aggregation of the output values of the fuzzy adapter.
Fuzzified values are written in a row vector: ms = $[\mu_{OP1}, \mu_{OP2}, \ldots, \mu_{OPn}]$.

example: membership functions around the operating points 5, 11 and 20

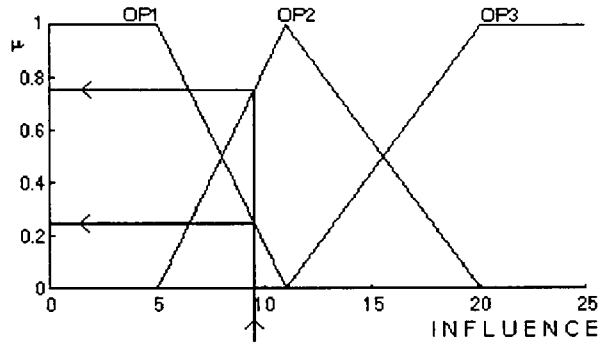


Figure 4-6: Membership Function, Example

The fuzzified input value 9.5 is written as the row vector
ms(9.5) = [0.25, 0.75, 0] in this example.

6) *The rule-bases for the different parameters are specified.*
Without limiting its general applicability, the Takagi-Sugeno model gains part of its efficiency from assuming singletons as output membership functions and fixing the implication and aggregation methods. For the suggested fuzzy-hybrid modelling approach, the same conventions are defined.

Using singletons as output sets, the rule bases can be summarised to very simple matrices. The single and dual variant cases are particularly easy to handle, yielding column vectors or two dimensional matrices, respectively, as rule bases. Each parameter of the system equation requires a separate rule-base matrix which is simply an ordered collection of the crisp values that the particular parameter takes on for the known operating conditions. The matrices are therefore directly taken from the equations in the third step of this sequence. The order of the matrix elements must be consistent with the membership row vectors - in a two-input case with

the columns of the rule matrices reflecting the parameter changes due to the first input ('A') to the fuzzy adapter and the rows reflecting the changes due to the second input ('B') and increasing matrix element indices referring generally to increasing absolute operating point levels of the inputs to the fuzzy adapter.

The structure of the rule-base matrix for any parameter "xy" which is dependent on the values of the process parameters A and B is:

$$RBxy = \begin{bmatrix} r_{A1B1} & r_{A1B2} & \cdots & r_{A1Bm} \\ r_{A2B1} & r_{A2B2} & \cdots & r_{A2Bm} \\ \vdots & \vdots & & \vdots \\ r_{AnB1} & r_{AnB2} & \cdots & r_{AnBm} \end{bmatrix}$$

This rule-base matrix is the short format of the rule-base

IF $A == A1$ AND $B == B1$ THEN $xy = r_{A1B1}$
IF $A == A2$ AND $B == B1$ THEN $xy = r_{A2B1}$
•
•
•
IF $A == An$ AND $B == Bm$ THEN $xy = r_{AnBm}$

Each element r_{AiBj} of the rule-base matrix represents therefore a rule. The rules have all the same weight (= 1) and the antecedents of a rule can only be combined by 'AND'. Thus, different antecedents that yield the same result (and could normally be combined by 'OR') must be put into separate rules.

example: In the simple, single variant case considered here as an example, the rule-bases are merely 3x1 - column vectors with their elements (i.e., parameter values) ordered according to the operating points.

$$RBb_0 = \begin{bmatrix} 10 \\ 6 \\ 15 \end{bmatrix}, \qquad RBa_1 = \begin{bmatrix} 1.2 \\ 4 \\ 7 \end{bmatrix}, \qquad RBa_2 = \begin{bmatrix} 9 \\ 25 \\ 0 \end{bmatrix}.$$

7) *Fuzzy implication, aggregation and defuzzification.* These steps take full advantage of the simplifications suggested in steps 5) and 6) in that they can be combined into a single, simple

and computationally efficient matrix operation. Using multiplication as implication method and aggregating simply the singletons, the multiplication of the membership vectors ("*ms..*") with the rule-base matrices ("*RB..*") combines all three steps. For a fuzzy adapter with two inputs, '*A*' and '*B*', this multiplication is carried out for all k parameters of the system equation as follows:

$$ms_A(x_A 0) \times RB_k \times ms_B(x_B 0)^T =$$

$$\begin{bmatrix} \mu_{A1} & \mu_{A2} \cdots\cdots & \mu_{An} \end{bmatrix} \times \begin{bmatrix} r_{A1B1} & r_{A1B2} & \cdots & r_{A1Bm} \\ r_{A2B1} & r_{A2B2} & \cdots & r_{A2Bm} \\ \vdots & \vdots & & \vdots \\ r_{AnB1} & r_{AnB2} & \cdots & r_{AnBm} \end{bmatrix}_k \times \begin{bmatrix} \mu_{B1} \\ \mu_{B2} \\ \vdots \\ \mu_{Bm} \end{bmatrix}$$

(the row vector $ms_B(x_B 0)$ must be transposed, (^{T});
$x_A 0$, $x_B 0$ are any input values to the fuzzy adapter)

example: The input value 9.5 is fuzzified to ms(9.5) = [0.25, 0.75, 0]; the transfer function parameters for this operating condition are calculated by multiplying the membership row vector ms(9.5) with the rule-base matrices.

$$b_0(9.5) = ms(9.5) \times RBb_0 = \begin{bmatrix} 0.25 & 0.75 & 0 \end{bmatrix} \times \begin{bmatrix} 10 \\ 6 \\ 15 \end{bmatrix} = \underline{\underline{7.00}}$$

$$a_1(9.5) = ms(9.5) \times RBa_1 = \begin{bmatrix} 0.25 & 0.75 & 0 \end{bmatrix} \times \begin{bmatrix} 1.2 \\ 4 \\ 7 \end{bmatrix} = \underline{\underline{3.30}}$$

$$a_2(9.5) = ms(9.5) \times RBa_2 = \begin{bmatrix} 0.25 & 0.75 & 0 \end{bmatrix} \times \begin{bmatrix} 9 \\ 25 \\ 0 \end{bmatrix} = \underline{\underline{21.00}}$$

8) *Transfer of the new parameter values to the linear parametric system equation and evaluation of the updated equation.* For an incrementally continuous simulation, this step, together with the matrix operation of step 7 and the fuzzification of the input signals can easily be programmed in any simulation environment and evaluated at fixed time increments.

example:

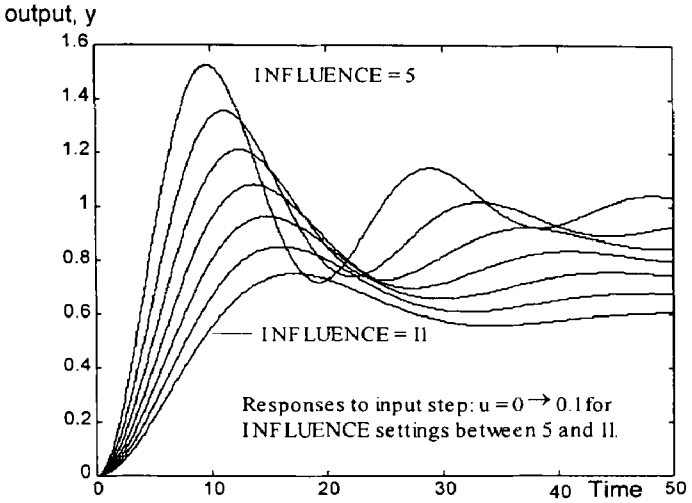


Figure 4-7 : Step Responses Of The Fuzzy Adapted Transfer Function At Different Settings Of 'INFLUENCE' Between The Initially Known Levels 5 And 11.

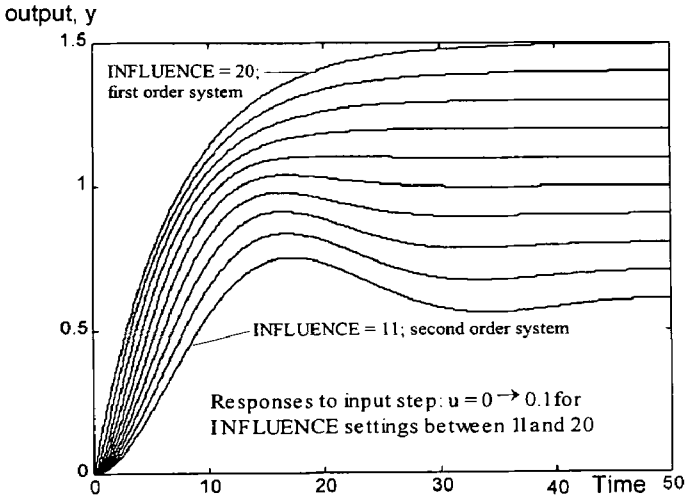


Figure 4-8: Step Responses Of The Fuzzy Adapted Transfer Function At Different Settings Of 'INFLUENCE' Between The Initially Known Levels 11 And 20.

The simulation results that are shown in Figure 4-7, Figure 4-8 and Figure 4-9 illustrate some of the important properties of the suggested approach:

- a continuous frequency shift in the oscillations of the second order responses between the initially specified parameter levels of INFLUENCE = 5 and 11
- the transition from second to first order behaviour in Figure 4-8
- the multivariability, with dynamic responses to changes of both the input, 'u', and the influence parameter ('INFLUENCE') in Figure 4-9.

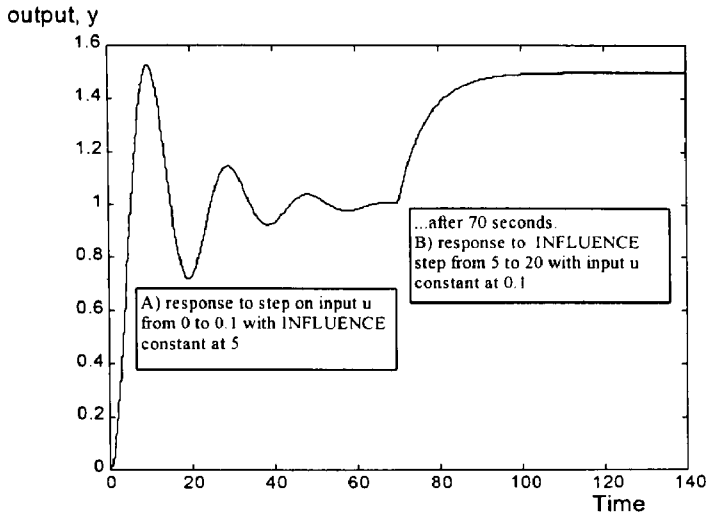


Figure 4-9: Successive Responses Of The Fuzzy Adapted Transfer Function To A Step On Input 'u' And On The Parameter 'INFLUENCE'.

Using the introduced simplifications as default settings for an automated approach to fuzzy hybrid modelling, all the user needs to supply is a set of operating points with associated locally valid SISO system equations. The surprisingly simple nonlinear multivariable modelling approach for practitioners is therefore based on both the suggested fuzzy hybrid modelling as such and the default settings with respect to input and output membership functions, implication, aggregation and defuzzification methods and the simple matrix notation.

4.3.2 The Proposed 'fuzTF' Approach Versus The Takagi-Sugeno Approach

Being based on locally valid system equations and using a fuzzy hybrid concept, the nonlinear process modelling approach by Takagi and Sugeno [103] is particularly closely related to the approach that is proposed in this chapter. Although structurally not quite as elegant and computationally not as efficient, the concept of the heterogeneous control system approach described by Kuipers and Aström [116] is conceptually the same as Takagi-Sugeno's.

As discussed above, both the Takagi-Sugeno approach and Kuipers-Aström's are based on averaging the output values of the locally valid system equations to obtain global results. While this kind of interpolation is a correct approximation for static relationships, averaging dynamic trajectories is theoretically improper. Although under certain circumstances the weighted interpolation of trajectories can yield good results, the limitation becomes particularly apparent if oscillating signals are considered: In the case that, for example, the system response to an

oscillating input signal is a sinusoid signal whose phase angle depends on another influence parameter, the local trajectories cancel each other to some extent, possibly even fully (180° phase difference), when they are averaged to the global output.

In Figure 4-10, an intermediate system response of the example from section 4.3.1 is determined according to Takagi-Sugeno (or equivalently to Kuipers-Aström) and shown together with the results of the approach introduced in this paper. While the Takagi-Sugeno model follows in terms of frequency and damping only the stronger oscillations of the trajectory for the pre-defined local model at INFLUENCE = 5, the suggested model shows correctly an intermediate gain, damping and frequency of the trajectory (see also Figure 4-7).

Since oscillations cannot be avoided in most dynamic simulations and also because the validity of process models should not depend on the type of input signal, the Takagi-Sugeno and Kuipers-Aström models should be built exclusively from static equations, which relate only instantaneous process variables.

Applying the Takagi-Sugeno approach to the combination of dynamic system equations (e.g. [99]) is, as opposed to the proposed 'fuzTF' approach, therefore "logically inconsistent", according to Murray-Smith's general verification considerations for system simulation models [118].

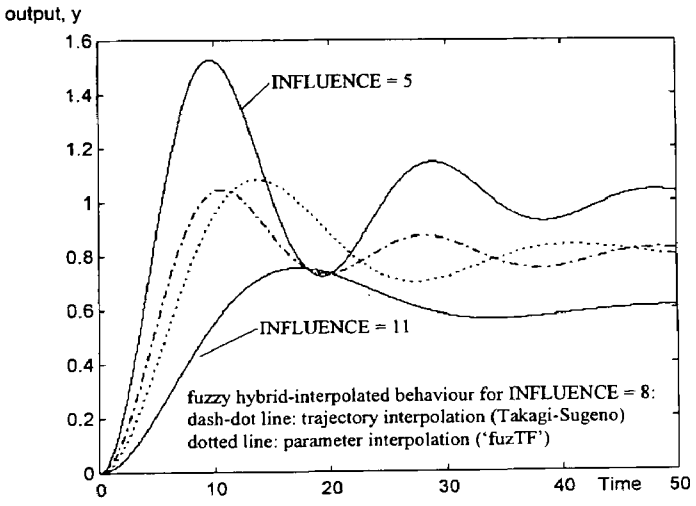


Figure 4-10: Step Response At Intermediate Level According To Takagi-Sugeno And 'fuzTF'

4.3.3 The Standard Structure Of The Global Fuzzy Hybrid Model

Global fuzzy hybrid process models consist, apart from the fuzzy adapter, of two modules (Figure 4-11): a block with the dynamic part of the parametric, fuzzy adapted transfer function ('fuzTF') and a preceding fuzzy static characteristic ('fuzSC'). The parametric fuzSC-function has the standard equation structure of a straight line:

$$Y = Kp \cdot U + Yo \qquad \text{or} \qquad Y = b_0 \cdot U + Yo$$

with

U = main input to the overall model; global value
Y = global steady state output
Kp = proportional gain in 'physically meaningful' terminology
b_0 = proportional gain in polynomial terminology
Yo = offset of the straight line

For the specification of the rule base RBYo, the offset Yo is calculated as

$$Yo = Y - Kp \cdot U \qquad \text{or} \qquad Yo = Y - b_0 \cdot U$$

from the known operating points (U,Y) and the gain Kp (or b_0) of the associated local transfer function.

The tangency of the straight line to the n-dimensional surface (n = number of influences fed into the fuzzy adapter + 1) of the static characteristic is achieved by the parameter adaptation through the fuzzy module. For each operating point, the 'fuzSC' represents therefore the locally valid, linearised static relationship and is thus the static equivalent of the dynamic 'fuzTF' relationship.

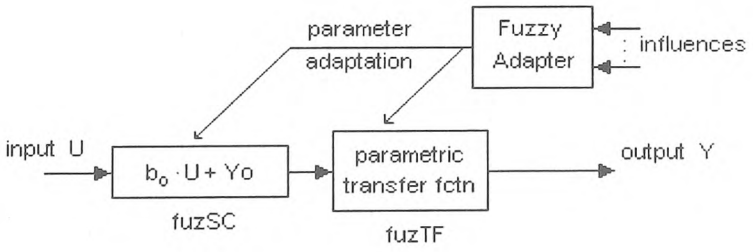


Figure 4-11: Standard Structure Of The Global Fuzzy Hybrid Model

Using the 'Hammerstein-like' structure of the fuzzy static characteristic preceding the dynamic block [109] as the ***standard structure of the global fuzzy hybrid model***, it is possible to accommodate different conventional model structures. The static characteristic of the considered funnel tank in section 4.4, for example, would have to be modelled conventionally by a 'Wiener' structure with the static characteristic following the dynamic block. The fuzzy hybrid model, however, can represent the real process behaviour in its standard structure simply by using the overall output as an influence parameter to the fuzzy adapter. A conventional 'Hammerstein' structure would be modelled in fuzzy hybrid terms by applying the overall input as an influence parameter for the fuzzy adaptation. Additionally, the standard fuzzy hybrid structure accommodates far more complex structures than 'Hammerstein' or 'Wiener', if any influence parameter - or even several of them - other than the overall input or output is used.

4.4 An Application Example

To illustrate the above listed advantages of the suggested modelling approach, this section gives an application example for a real process. Another important purpose of this section is the external validation of the 'fuzTF' approach. The simulation results are therefore compared with the ideal process behaviour.

A water tank in the shape of a funnel (Figure 4-12) is considered here. This tank behaves very nonlinearly with respect to its gain and time constant due to the outflow equation (square root function) and the variation of the tank area over its height. The chosen main process input is the water flow into the tank, Q_{in} [m^3/s] and the output is the water level, h [m]. Additionally, the varying outlet area A [m^2], which is operated by a valve, is considered as a secondary process input.

The funnel tank is a good demonstration example for the application of the modelling approach in process industry:

- It behaves very nonlinearly.
- It is multivariable. Initially, the process could be modelled with constant outlet area and later it could be extended to cover varying outlet areas (stepwise model extension). In this presentation, however, the complete process is modelled in one step.
- The process is stable and therefore enables the use of very basic open-loop identification approaches.
- Typical for process industry, the time constants are big enough to allow for the application of an 'intelligent' modelling interface that aims at translating the user's process experience into a model (cf. Chapters 5 and 6).

As opposed to the funnel tank, unstable and fast test processes like the inverted pendulum are not appropriate systems to show the validity of this nonlinear multivariable modelling approach, because they do not address its particular purpose. This is not to say that the introduced model could not handle such processes. However, it is important to stress again that the goal and uniqueness about the proposed model is not the time-critical performance in simulation experiments, but the ease with which a very satisfactory, highly nonlinear, multivariable process model can be derived from quite basic information that can quickly be obtained for stable processes - even in an industrial environment and by engineers that are not modelling experts.

A direct comparison with other modelling approaches that are based on partial process information is not shown here, because these are either based on very different assumptions with respect to the available modelling information, or are - like the Takagi-Sugeno or Kuipers-Aström approaches - inapplicable to processes with varying dynamic characteristics (e.g. time constants, damping ratios), as discussed and illustrated in section 4.3.2. However, the simulation results of the fuzzy hybrid model are compared with the theoretically derived process model, which represents in this case anyway the 'ideal' behaviour.

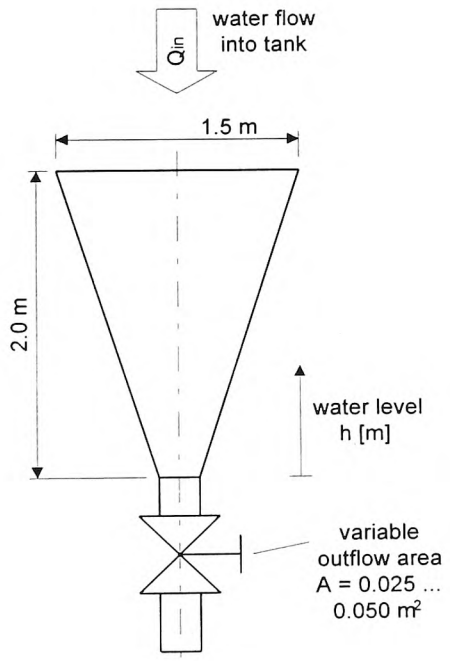


Figure 4-12: The Funnel Tank

The modelling sequence:

1) In this particular example, the system **output** (water level) influences the dynamic behaviour of the real process and is therefore also in the model fed back as the **input** to the fuzzy module which adapts the parameters of the transfer function. Thus, the water level, h, is the first input to the fuzzy adapter. It is important to note that this exceptional situation of an internal feedback in the model imposes a particular challenge to the quality of the global process model and its simulation results. The external influence 'outlet area' is an additional input to the fuzzy adapter.
2) a) Four different water levels were considered: h_1 = 0.131m, h_2 = 0.566m, h_3 = 1.094m and h_4 = 1.616m, which are the steady state levels that resulted from the small perturbations of the process to determine the local transfer functions.
   b) Three outlet areas are distinguished: 0.025m^2, 0.035m^2 and 0.050m^2.
3) From the local step responses[3] (first order proportional behaviour), the characteristic gains and time constants were determined as follows (Table 4-1), using the simple tangent evaluation at t = 0 :

	A_1 = 0.025m^2	A_2 = 0.035m^2	A_3 = 0.050m^2
h_1 = 0.131m	$G_{11}(s) = \frac{5.7}{0.39s+1}$	$G_{12}(s) = \frac{4.3}{0.36s+1}$	$G_{13}(s) = \frac{3.1}{0.32s+1}$
h_2 = 0.566m	$G_{21}(s) = \frac{12.8}{2.62s+1}$	$G_{22}(s) = \frac{9.3}{2.29s+1}$	$G_{23}(s) = \frac{6.6}{1.88s+1}$
h_3 = 1.094m	$G_{31}(s) = \frac{18.1}{8.47s+1}$	$G_{32}(s) = \frac{13.1}{7.08s+1}$	$G_{33}(s) = \frac{9.4}{6.34s+1}$
h_4 = 1.616m	$G_{41}(s) = \frac{22.1}{20.14s+1}$	$G_{42}(s) = \frac{16.0}{16.28s+1}$	$G_{43}(s) = \frac{11.6}{12.87s+1}$

Table 4-1: Local Transfer Functions

4) The parametric system equation which is valid throughout the operating range is therefore:

$$G(s) = \frac{K_p}{T \cdot s + 1}$$

Using this structure, the resulting fuzzy hybrid model can predict very well any local changes to any intermediate level within the operating range. It is therefore applicable to many closed loop simulations. In this example, however, the aim was to build a global model that predicted the output, or water level, within the operating range in absolute terms on the basis of the absolute

[3] These local step responses resulted from simulating the theoretically derived mathematical model. The fuzTF model is therefore deduced from the local behaviour of this theoretical model. Hence, the trajectories of the theoretical model represent the 'ideal' behaviour with which the fuzTF results must be compared for validation purposes.

input signal (Q_{in}). For this purpose, the adapted gain Kp is considered together with a likewise adapted offset Yo (Table 4-2) of the water level in a separate static equation according to the standard structure of the global fuzzy hybrid model (Figure 4-13).

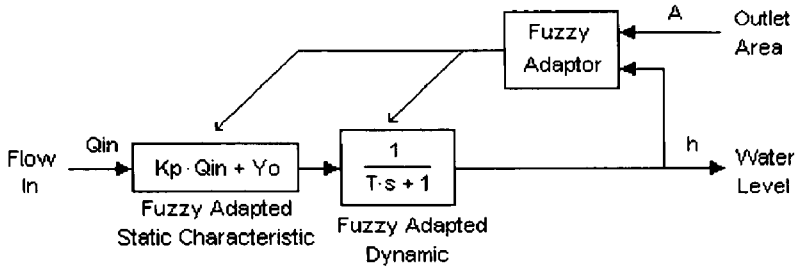


Figure 4-13: Fuzzy Hybrid Model Of The Funnel Tank With Variable Outlet Area

All required process parameters are summarised in the following Table 4-2.

A [m^2]	h [m]	Q_{in} [m^3/s]	Kp	Yo *)	T
0.025	0.131	0.0400	5.7	-0.097	0.39
	0.566	0.0833	12.8	-0.500	2.62
	1.094	0.1158	18.1	-1.002	8.47
	1.616	0.1407	22.1	-1.494	20.14
0.035	0.131	0.0560	4.3	-0.110	0.36
	0.566	0.1166	9.3	-0.518	2.29
	1.094	0.1621	13.1	-1.030	7.08
	1.616	0.1970	16.0	-1.536	16.28
0.050	0.131	0.0800	3.1	-0.117	0.32
	0.566	0.1665	6.6	-0.533	1.88
	1.094	0.2315	9.4	-1.082	6.34
	1.616	0.2814	11.6	-1.648	12.87

Table 4-2: Summary Of The Process Parameters

*) $Yo = h - Kp\ Q_{in}$

5) The default triangular membership functions with $\Sigma\mu_i = 1$ are applied here for both influences. The peaks of the fuzzy sets ($\mu = 1$) are set at the distinguished water levels and outlet areas given in 2):

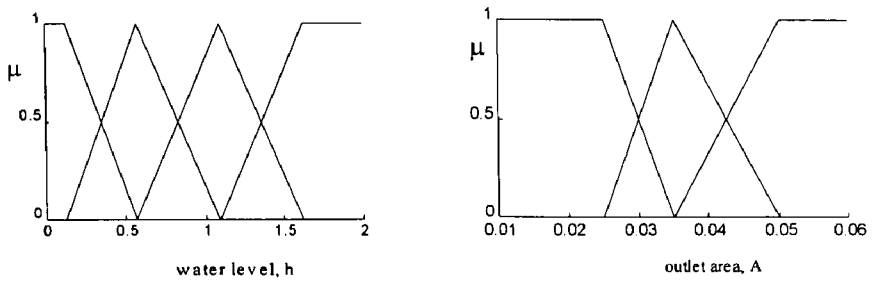


Figure 4-14: Membership Functions For The Funnel Tank Model

6) The rule bases are defined as follows:

$$RBKp = \begin{bmatrix} 5.7 & 4.3 & 3.1 \\ 12.8 & 9.3 & 6.6 \\ 18.1 & 13.1 & 9.4 \\ 22.1 & 16.0 & 11.6 \end{bmatrix} \qquad RBYo = \begin{bmatrix} -0.097 & -0.110 & -0.117 \\ -0.500 & -0.518 & -0.533 \\ -1.002 & -1.030 & -1.082 \\ -1.494 & -1.536 & -1.648 \end{bmatrix}$$

$$RBT = \begin{bmatrix} 0.39 & 0.36 & 0.32 \\ 2.62 & 2.29 & 1.88 \\ 8.47 & 7.08 & 6.34 \\ 20.14 & 16.28 & 12.87 \end{bmatrix}$$

7) As described in section 3, the fuzzy implication, aggregation and defuzzification were simply summarised by multiplying the fuzzified inputs to the fuzzy adapter (i.e., the membership vectors of the water level and the outlet area) at each simulation increment with the rule base matrices, yielding updated parameter settings for Kp, T and Yo.
8) The updated parameters are transferred to the static and dynamic system equations which are evaluated in the new continuous simulation increment.

Using the physical balance equations, a process model of the funnel tank was derived theoretically (Figure 4-15). This model was simulated concurrently with the fuzzy hybrid model in the validation tests and served as a direct reference.

The signals for the simulation (cf. Figure 4-16, Figure 4-17 and Figure 4-18) were defined in particular to explore the model behaviour in intermediate areas between the previously known points. The particular shapes of the signals in Figure 4-16 and Figure 4-17 are quite meaningless for the real funnel tank but serve the purpose of investigating how well the fuzzy hybrid model (dotted line) handles rather more 'awkward' situations in comparison with the ideal behaviour (full line).

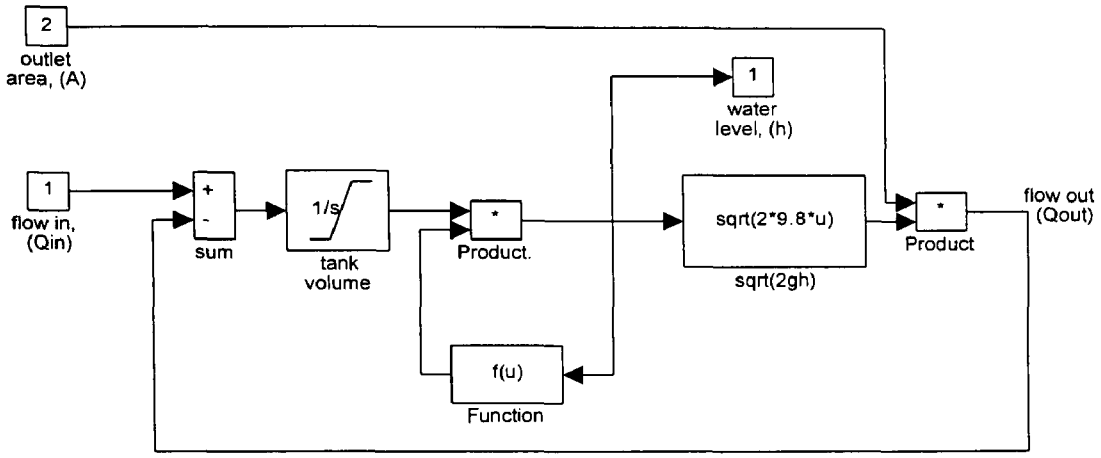


Figure 4-15: The Theoretically Derived MIMO Funnel Model

Definition of the Function f(u) with "u" being here the current water level:

f(u) = 12/(π*((D/(ht+ht*d/(D-d))*(u+ht*d/(D-d)))^2+(D/(ht+ht*d/(D-d))*(u+ht*d/(D-d)))*d+d^2))

D = top diameter of funnel tank

d = bottom diameter of funnel tank

ht = height of funnel tank

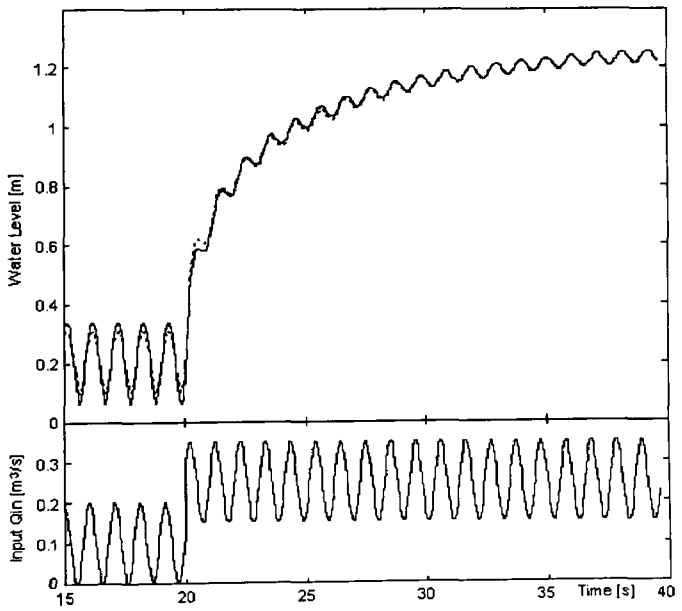


Figure 4-16: Simulation Results Of The Fuzzy Hybrid (Dotted Line) And The Theoretically Derived Funnel Model (Full Line) At A = 0.05 m^2

The sinusoid input signal with constant frequency and amplitude but changing offset (Figure 4-16) is a very severe test to the fuzzy adapter - in particular during the transition from low to high water

level. The fact that the fuzzy hybrid signal follows the ideal response so well with respect to gain and phase indicates the successful continuous adaptation of the *dynamic* properties.

It is, however, especially interesting to see in Figure 4-17 the fuzzy model responding dynamically to the variations of the outlet area 'A', since information with respect to the dynamic relationship h = f(A) was not directly given in the modelling process (the dynamic process information was limited to the main h = f(Q_{in}) relations at certain static settings of the influence parameters). The positive simulation results illustrate therefore how a simple parametric transfer function is 'promoted' to a multivariable dynamic process model by tuning its parameters according to the influences.

Although the water level is, like the outlet area, an influence to the dynamic process behaviour, it is, as opposed to A, not an additional input to the overall model, but an internal influence that is fed back. It has therefore more the characteristic of a dynamic nonlinear influence.

Bearing in mind that the global (nonlinear and multivariable) fuzzy hybrid model was only derived from local, linear SISO (single input - single output) process information, the model performs very well in the direct comparison with the 'original' process behaviour. Apart from the relatively little process information and the challenging test signals, the internal feedback of the model output as an influence imposes a particular difficulty. The output of any model is always an approximation of the real output - in the funnel tank example, the output is therefore fed back together with its model error and re-used for the calculation of updated parameters. This bears the risk of an accumulating prediction error, and wherever possible, such internal feedbacks should therefore be avoided.

Inevitably, the consideration of an increasing number of input (or influence) variables in the process model results also in the combination of uncertainties and interpolation errors, as the comparison of Figure 4-17 ($\mathbf{Q_{in}}$, **A** and **h** varying at the same time) with Figure 4-16 and Figure 4-18 (A = constant) shows. Nevertheless, the application of multivariable process models is normally of great advantage to the simple neglection (i.e., linearisation) of influences, even if the available process information is quite basic, as the example shows.

Overall, the fuzzy hybrid model performs very well, without the tendency of an increasing output error and representing even the phase and amplitude changes during transition periods as well as the effects of concurrent variations of all three inputs/influences (Q_{in}, A and h) correctly.

For the additional validation of the 'fuzSC' interpolation qualities, please refer to Appendix A4, where the fuzzy static characteristic is applied to different benchmark problems.

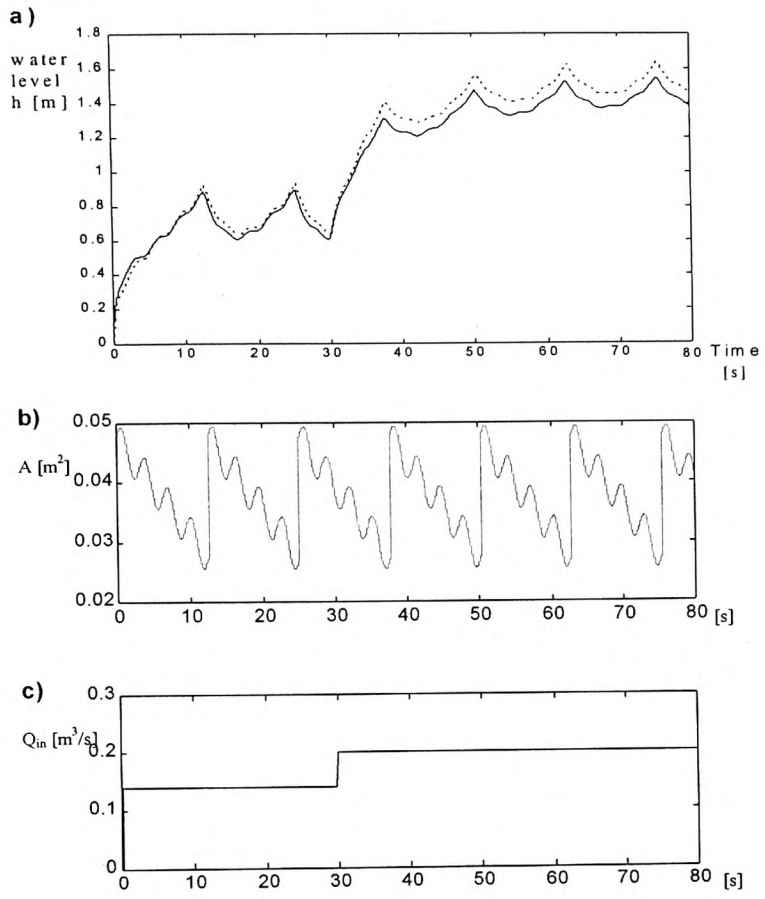


Figure 4-17: a) - Simulation Results Of The Fuzzy Hybrid (Dotted Line) And The Theoretically Derived Funnel Model (Full Line); b) And c) - The Related Input Signals That Cause The Above Response a)

4.5 Summary: The Fuzzy Hybrid Approach And Its Properties

Like the Takagi-Sugeno model, the suggested approach is suited to process identification on the basis of measured data. In fact, the results of the structure and parameter identification steps suggested by Takagi-Sugeno [103] and Sugeno-Kang [99] could also be used with the fuzzy-hybrid model which has been introduced in this paper. Rather than trying to identify the fuzzy hybrid model directly from process data, the main aim of this approach, however, is the integration of process information in the form of locally valid, basic system equations that can easily be obtained even in an industrial environment, using either an easy-to-use linear identification tool or a knowledge acquisition approach that translates process experience into linear transfer functions (cf. Chapter 5 and [107, 108]).

Both this and the Takagi-Sugeno approach benefit from simplifications that make the algorithms more efficient; the simplifying assumptions of both approaches are largely the same (e.g. singletons

as output membership functions, multiplication as implication method).

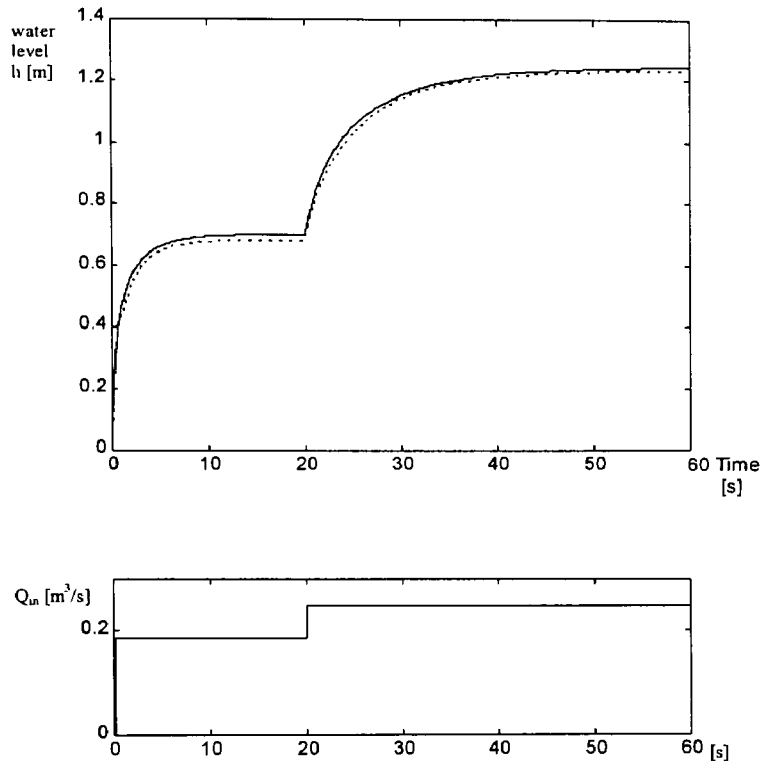


Figure 4-18: Global Step Responses At The Two Intermediate (i.e. Previously Not Defined) Steady State Levels h = 0.7m And 1.25m (A = 0.05 m^2)

The Takagi-Sugeno approach further benefits from the low number of rules required and from the fact that the linear equations form an integral part of the fuzzy model, while the introduced approach necessitates in the same modelling situation the number of varying parameters times the number of rules of Takagi-Sugeno's model. However, the rule-bases in the proposed approach are merely simple matrices which are directly read from the local equations and the implication, aggregation and defuzzification are extremely efficient matrix operations. Instead of evaluating a separate system equation for every rule, only a single equation with the adapted parameters is evaluated. Nevertheless, the more 'integrated' characteristic of Takagi-Sugeno's model which does away with the actual 'adaptation' of a somewhat separate component is still advantageous for the combination of piecewise linear *static* equations. Still though, when it comes to combining locally valid *dynamic* equations to an overall model, there is no way around the calculation of a single overall output trajectory in the general case. Therefore, the approach suggested in this chapter appears to be a sensible extension of fuzzy hybrid modelling towards dynamic system modelling and thus a complement to Takagi-Sugeno's model for static relationships.

The introduced modelling approach has been developed in order to overcome the shortcomings of fuzzy modelling that have been pinpointed in the previous chapter. Within the overall context of this work, the approach addresses the industrial engineer's need of simplified nonlinear multivariable modelling on the basis of partial process information. Particular advantages of the suggested fuzzy-hybrid modelling approach compared with existing modelling techniques that are based on partial process information are as follows:

Most notably, the approach as such, the modelling sequence and the resulting model are exceptionally simple:

- The model structure is easy to understand - even with very little fuzzy logic knowledge.
- The fuzzy hybrid model has a standard block structure: for global process models, the fuzzy static characteristic *always* precedes the fuzzy adapted dynamic block (Hammerstein-style), even if the process would have to be modelled conventionally using a Wiener- or other, more complex structures.
- The model can be easily programmed in a general purpose simulation environment.
- Since it takes advantage of the suggested simplified standard membership functions and implication, aggregation and defuzzification methods as default settings, the modelling approach can be largely automated, hiding in particular all parts of the fuzzy approach, if required (cf. Chapter 6). The defaults should only be edited by more experienced modellers.
- Once automated, the effort for building nontrivial models is extraordinary small.
- The only required modelling information is a set of operating points with associated locally valid SISO system equations which can be easily acquired using simple identification approaches (cf. [107]).

This simplicity of the approach is well in line with the general idea behind modelling. Since the model-behaviour is always only an approximation of the real behaviour, the model must be kept as simple as possible for a specific application [52]. The suggested approach follows therefore this philosophy as an extension of simple linear dynamic modelling towards modelling a class of nonlinear dynamic processes.

Despite its simplicity, however, the modelling approach is powerful in that it can combine the local dynamic process knowledge to highly nonlinear, multivariable global models. The particular benefit for practical users is therefore the availability of an approach that enables the modelling of such complex processes despite the various constraints in industry that have been discussed in the introduction.

The quality of a model of the suggested type is mainly dependent on the quality of the local system equations as well as the partition of the input space. The approach as such is therefore as applicable to coarse modelling on the basis of assumptions, separating, for example, only two characteristic operating conditions, as it is to the combination of several well determined local system equations for precise predictions of the process behaviour.

Unlike conventional parameter interpolation approaches, the fuzzy interpolation covers also multiple valued variations like hystereses or even n-dimensional, multiple variable relationships within the same concept (cf. the example in section 4.4 and Appendix A4).

A very useful aspect is that conventional analysis techniques (e.g. stability) can in parts be applied.

Further, it should be noted, that the suggested approach is in principle not limited to the combination of piecewise linear dynamic system equations but could likewise integrate some types of nonlinearities that change under different process conditions. An example of particular practical importance is varying dead-times. Although our current investigations are focused on piecewise linear systems, an extension towards some nonlinearities is merely an implementational question.

Another major advantage of the suggested modelling approach is its modularity that helps to keep the model complexity and modelling effort to a minimum: after considering initially only the most important influence on the nonlinear behaviour of the process, further influences can be added to the model at a later stage if required. Likewise, the model quality can be improved in a stepwise manner by adding intermediate operating conditions to the rule bases and membership functions. Previous versions of the process model can always be re-used in such situations.

The unique advantages of the suggested fuzzy hybrid approach as well as the combination of these characteristics make it particularly useful for the modelling of dynamic large scale systems in industry.

The fuzzy hybrid modelling approach was considered in the design of the knowledge acquisition sequence which is described in the following chapter. By making the fuzzy hybrid modelling an integral part of the implementation of the knowledge engineering approach, the generation of appropriate simulation models - including the standard block structure and the rule base matrices - was fully automated. The knowledge acquisition interface, which is introduced in Chapter 6, enables therefore a new way of simple and fast nonlinear dynamic and multivariable modelling on the basis of experience which is particularly geared at industrial requirements.

5. *MODEL*^{ing} - The Knowledge Engineering Approach

The importance of developing a structured procedure for the knowledge acquisition - the "Knowledge Engineering Approach" - was pointed out in Chapter 2. After the decision for the fuzzy notion as the most suitable approach to model processes on the basis of partial knowledge and proposing a new fuzzy hybrid approach to address nonlinear system dynamics, this key point from Chapter 2 is taken up again in this Chapter. The "Knowledge Engineering Approach" is described here together with the considerations that led to its particular design. Special care has been taken to incorporate the efficient handling of complex dynamic systems according to the "fuzTF" concept that was detailed in the previous Chapter. This design consideration ensures a smooth integration of the two main contributions of this work.

In the first two sections of this Chapter, preparatory steps are described: the decision on the knowledge representation format and the selection of an analysis and design technique. Afterwards, the design of the knowledge elicitation procedure as such is detailed.

5.1 Knowledge Representation

For the design of an approach that acquires and handles the process information of its user, it is essential to decide on the representation format of this information - or knowledge - within the system. A particularly important aspect of this consideration is the need to *structure* the, previously largely unstructured, knowledge in order to facilitate efficient knowledge handling and communication to the control and modelling experts. Knowledge structuring is, in fact, one of the key issues in knowledge acquisition [45].

Since the amount of acquired knowledge varies from case to case, this knowledge representation should be flexible so that it adapts to each situation.

The most important knowledge representations in system science are:

Mathematical knowledge representations:
Mathematical system descriptions can be derived on the basis of theoretical analyses of the physical properties and laws that apply to systems or from the identification of input / output data. The results can take the form of:

- **Algebraic** system definitions - difference equations for continuous or discrete sampled processes; possibly in state space format
- **Analytical** system definitions - differential equations for continuous processes, possibly in state space format)
- **Logic based** system descriptions - e.g. predicate logic, for the description of discrete processes.

Rule-based knowledge representations:

Knowledge that can be formulated into statements of the form "IF <condition> THEN <consequent>", can be handled by rule-based knowledge representations.

- **Crisp rules** - IF ... THEN ... rules, as applied in conventional expert systems, using forward or backward chaining for applications like simulation or fault diagnosis, respectively.
- **Fuzzy rules** - IF ... THEN ... rules with additional handling of imprecision

Object oriented representations:

Object oriented representations handle knowledge that is or can be quantized into discrete, distinguishable entities called objects. Apart from real world objects such as a controller, conceptual entities such as control strategies can be objects, too.

- **Semantic networks** - the labels of real world or conceptual objects as well as properties and characteristics are linked via multiple relationships, illustrated by arrows, to complex networks.
- **Frame-based** representation - hierarchically structured classes of real world or conceptual, i.e., abstract objects with the associated properties and procedures directly attached to the objects.

Connectionistic representations:

Connectionistic representations are based on knowledge in the form of input / output data of the considered system that is used to train the connectionistic structure so that it represents the functional relationship of the system or to classify input data in a graphical map.

- **Neural networks** - relation of multiple data inputs to multiple data outputs via a layered network of neurons, i.e., elementary units that determine an output signal on the basis of its weighted input signals and pass it on to the next layer of neurons.
- **Associative Memories** - similarly to neural networks, associative memories map n-dimensional input spaces to m-dimensional output spaces, yet using an approach that is based on the idea to split the input space into regular n-dimensional sections and associate an output vector to each of these sections.
- **Feature Maps** - classify n-dimensional input data into locations within a mostly 2-dimensional map, which is like the associative memories structured into a grid, yet in a non-equidistant fashion

All these knowledge representations have advantages and disadvantages with respect to each other. There is, therefore, no such thing as "the best" representation. The major difference between these notions is the required knowledge format, which is therefore also the key to selecting an appropriate representation:

Knowledge, in the context of this work can take on various formats; a positive selection of one of the knowledge representations is therefore not possible. Hence, the limitations of each of the techniques is considered instead:

- Mathematical knowledge representations are not *generally* applicable, because the envisaged user cannot normally provide theoretical mathematical information. Nevertheless, a facility to handle this kind of information would be sensible.
- Likewise, the rule-based knowledge representation is not *generally* applicable, because not all potential information (yet possibly a significant amount) could be formulated in rules.
- Connectionistic representations focus exclusively on numerical input / output data and are therefore outside the scope of this work (hence inapplicable).
- Object oriented knowledge representations have no particular drawback or limitation with respect to their application in the context of this work.

The object oriented knowledge representation is, in fact, a very versatile and powerful notion. Even an analytical system description, a rule-based process model or a neural network representation can be objects within the object oriented approach. This means that other knowledge representations can be integrated in the object oriented representation. Furthermore, all knowledge *types* (i.e., causal, heuristic, case-based and probabilistic knowledge[1]) can be accommodated, too. This versatility of the object oriented knowledge representation makes it ideally suited to the requirements of this work.

Within the object oriented category, it is clearly the frame-based approach that is advantageous. While semantic networks are highly complex and unorganised, frames provide a clear hierarchical structure and therefore fulfil the above stated requirement of a representation that facilitates an efficient knowledge structure.

[1] cf. Appendix A2

The Flexible Frame concept:

In order to address the required flexibility of the knowledge representation that adapts to the provided amount of information, it was decided to define a flexible frame as follows.

In the design of the object oriented approach, the different parts of possibly available process information must be separated in different classes. Whenever a particular piece of process information can be provided during modelling, an instance of the appropriate class is created and specific modelling results are passed to the prepared slots in that instance. The collection of instances related to the same component must be efficiently handled by associations. Creating only as many instances as the process information requires, the frame-based representation adapts always to the individual situation. By this means, big standard frames with a wealth of unused variables (i.e., 'sparse' frames) can be avoided.

This concept can be clarified by considering some of the basic object oriented terminology as follows:
Object oriented representations handle knowledge that is or can be quantized into discrete, distinguishable entities called ***objects***.
Each object has its own ***identity***, which means that two objects are distinct even if all their attribute values are identical.
Objects with the same data structure (***attributes*** or ***slots***) and behaviour (***operations***) are grouped into a ***class***. Each class can describe an infinite set of individuals, i.e., objects, which are called ***instances*** of that class.
Classes are often arranged in hierarchical structures. They share their operations and attributes with their ***subclasses*** on lower hierarchical levels. This sharing of properties is called ***inheritance***. Each subclass ***inherits*** all the attributes and operations of its ***superclass*** and is further characterised by its own additional properties. Inheritance exists only in the direction *from* superclass *to* subclass. At the end of the "branches" in this hierarchical inheritance "tree", the "leaves", i.e., the instances, inherit all properties from the class from which they are ***instantiated***. Instances as concrete entities cannot inherit any further.

These fundamental terms are sufficient to describe this work. Comprehensive introductions to object orientation as such and the particular terminology are given by Rumbaugh et al. [34] and Booch [119]. Jobling, et al. [23] put these aspects additionally in a control engineering context.

5.2 Selection Of A Design Technique

In order to systematically approach the problem of system design as well as to document the results of this process, a structured technique is required.

In this research project, the design of the knowledge engineering **methodology** is of central interest while the programmatic implementation serves mainly as a tool to prove the applicability and validity of the knowledge acquisition sequence and the modelling techniques. This emphasis must be considered in the selection of a structured design technique.

Information on the sequential flow of the methodology is of particular importance because

- the user must be put stepwise into the context of increasingly complex questions about the process so that the understanding is maintained and
- redundant questions must be omitted to keep the knowledge acquisition as short as possible - this is done by 'strategic' placing of key questions within the sequence and intermediate evaluation of partial information.

Hence, a primary requirement of the design technique is the facility to illustrate sequential flows and conditional decision processes.

When it comes to sequential flows, it is probably the Flow Chart that springs first into one's mind. The problems of applying flow charts for program design have been pointed out long ago. Jackson [120] illustrated with an example the difficulties of making amendments to programs that have been built as direct algorithmic implementations of flow charts.

> "Design (that is, program design) is about structure and flowcharts, as their name suggests, are about the flow of control. At the time when the designer should be thinking about the shape of his problem, the flowchart encourages him to think about execution of the program inside the computer." Jackson [120]

In the context of this work, this statement must be qualified in that it was referring to conventional computational problems rather than knowledge acquisition approaches. In the latter case, one of the main design problems **is** the flow of control (cf. Chapter 2). Still though, Jackson has a point: even for the problem considered here, it is important to lay out a general concept of the 'things' that are affected by the flow of control before this sequential flow itself is devised.

In any case, the traditional flow chart has further disadvantages. It is not very suitable for focusing on different hierarchical levels (i.e., nested charts) and therefore not useful for bigger problems; it is 'bulky', takes up a lot of space and it allows normally only for basic yes/no decisions.

The structured design technique exemplified by the work of Jackson focuses on program structuring according to the structure of the data to be processed (i.e., 'Data-Driven Design'). Other design approaches apply algorithmic decomposition and can therefore be categorised as 'Top-Down Design' techniques (e.g. De Marco, [121]). Sequential flows, however, are normally outside the scope of both categories of structured design notions.

Above, the 'things' that are affected by the flow of control, have been mentioned. These 'things' are actually a key point in our further search for an appropriate analysis and design technique since they are technically nothing else than *objects*. The advantages of object orientation have been briefly discussed in section 5.1 where it was decided to apply an object oriented knowledge representation. In order to benefit throughout the whole approach from the notion of object orientation, the focus is put on such design techniques.

'Object Oriented Design' (OOD) methods are best exemplified by the work of Booch [119]. In this method, systems are modelled as collections of co-operating objects, treating individual objects as instances of a class within a hierarchy of classes. The notation for object-oriented design includes four basic diagrams (class diagrams, object diagrams, module diagrams, and process diagrams), which are all static, and two supplementary diagrams (state transition diagrams and timing diagrams). The state transition diagrams, however, show only state changes within a single object and not among a set of collaborating objects while timing diagrams, which have their roots in hardware design, are mainly appropriate for scheduling absolute or relative lengths of time for different operations. The inclusion of PDL (program description language) annotations or simple numbering of messages within object diagrams are therefore the best - albeit unsatisfactory - means of documentation of sequential flows within Booch's OOD notion.

According to the **Object Modeling Technique (OMT)** by Rumbaugh et al. [34], the analysis and design of a system is handled by three different models, each of them focusing on a particular viewpoint. These three models are related to each other and can be considered as orthogonal views of the system:

- The **object model** represents the data aspects, i.e., the static, structural view.
- The **dynamic model** represents the control aspects; it illustrates the behaviour and temporal sequences of system states.
- The **functional model** represents the function aspects, i.e., data transformation and flow.

All three models are required for a complete system description.

For the design of the knowledge acquisition methodology, the advantages of OMT are clearly rooted in the dynamic model. It represents the sequential flow of control, including simple decision processes (y/n), multiple-decision processes (choice) and decisions to be made on the basis of previously given information (conditions). Also, the OMT diagrams can be efficiently nested and very flexibly arranged, making good use of the available space.

Furthermore, this technique is extraordinarily simple, yet powerful in comparison with the previously discussed system design techniques and it combines ideally the notion of object orientation with the design of the flow of control.

Another major advantage of OMT is its flexibility. The user of this structured design approach is explicitly encouraged to use only the modelling constructs that are needed for the given problem. And although all three models (object-, dynamic- functional model) are required for a complete system description, emphases can easily be achieved by going into more or less detail with each of them.

Therefore, the Object Modeling Technique was selected to devise the knowledge acquisition approach. It is acknowledged that this has not been a complete review of analysis and design techniques and not all of the considered approaches have been mentioned. However, since several publications have focused on comprehensive reviews and comparisons, the reader is referred to De Champeaux and Faure [122], Jobling et al. [23] and Rumbaugh et al. [34] as starting points.

Before the actual analysis and design of the *MODEL*^{ing} approach is documented in the following section, the three OMT models are explained in accordance with Rumbaugh et al. [34] in a little more detail:

The **Object Model** describes the structure of objects that makes up the considered system or problem domain. It defines the identity of the objects, their attributes and relationships amongst each other. **Object Diagrams** are the graphical representation of the object model. In the object diagrams, object classes are arranged into hierarchies sharing common structure and behaviour and are associated with other classes. Figure 5-1 gives a brief summary of the notation.

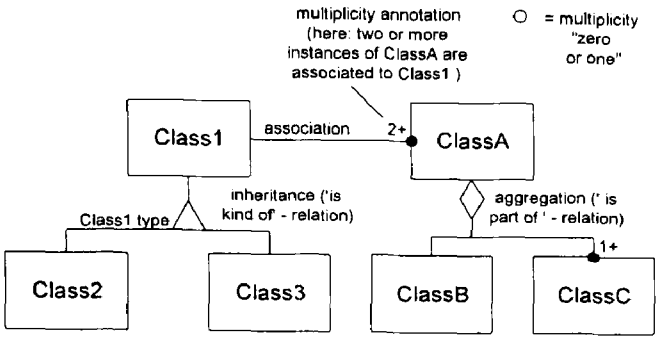


Figure 5-1: Notation For The Object Diagram

The **Dynamic Model** describes temporal aspects of a system, in particular the sequencing of operations. This includes the different system states and their sequential flow, events and sequences of events that mark the changes between the system states and the organisation of events and states. Hence, the dynamic model captures the 'control' aspect of the system; it is represented graphically with **State Diagrams**. Figure 5-2 summarises the notation that is applied in the next sections.

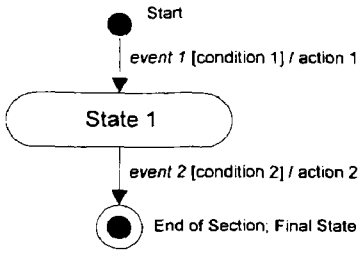


Figure 5-2: Notation For The State Diagram

The **Functional Model** describes all aspects of data transformation and flow. It captures what a system does, without regard for how or when it is done. **Data Flow Diagrams** are the graphical representation of the functional model. Without regard for when or if functions are executed, data flow diagrams show the dependencies between input, output and internal values of a system. Again, the basic notation that will be used in the following is summarised below in Figure 5-3.

The role of each of the three models defines their relationship: The functional model specifies what happens, the dynamic model specifies when it happens, and the object model specifies what it happens to.

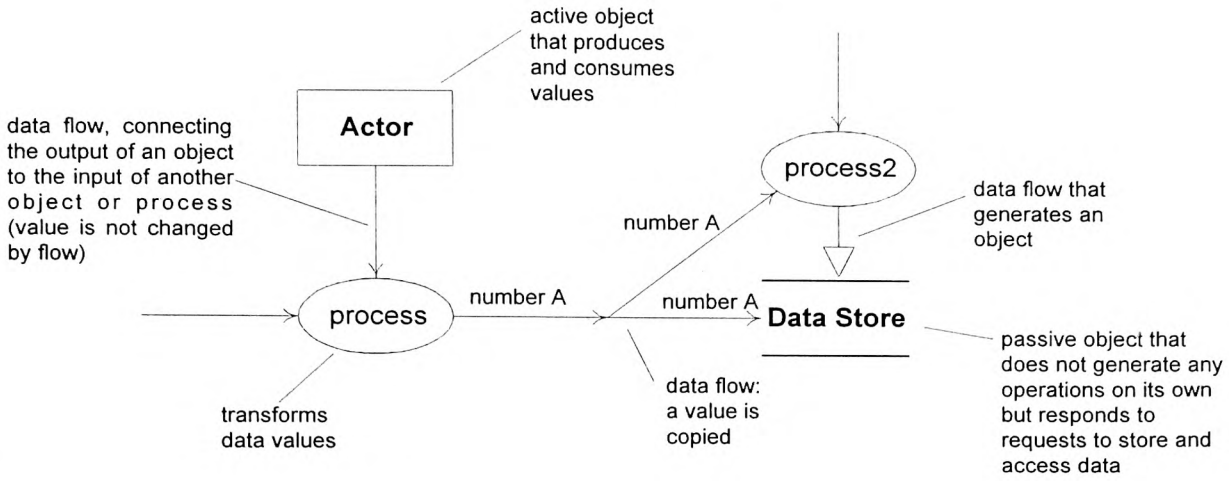


Figure 5-3: Notation For The Data Flow Diagram

Although this information should be sufficient to understand the complete design of the *MODEL*^{ing} methodology, the reader is referred to Rumbaugh et al. [34] for a detailed introduction to the Object Modeling Technique.

5.3 Analysis And Design Of The Object Model

After starting off with the documentation of the object model for the *MODEL*^{ing} approach in this section, the following sections 5.4 and 5.5 deal with the documentation of the dynamic- and functional model, respectively. In all these "analysis and design" sections, the focus is on *what* needs to be done, independent of *how* it is done.

Due to the variety of knowledge component types and the arbitrary number of individual pieces of such information, a flexible knowledge representation framework was considered as being particularly important, as was discussed above. Throughout the object modelling, care had to be taken to maintain as much openness as possible to enable later the design of the flexible knowledge representation on the basis of individually created, multiple instances per model.

After systematically analysing the domain of interest and deciding on the important objects to be considered, a first OMT-object model is normally built and refined until it appears to be an 'ideal' representation [34]. This first object model that results directly from the domain analysis (and likewise all other initial OMT diagrams) is 'ideal' in a sense that it appears to be the best solution, irrespective of the implementation. The implementational considerations normally necessitate some

revisions of the initial conceptual design. Figure 5-4 shows this 'ideal' object model for the knowledge engineering approach "*MODEL*^{ing} ":

The three main classes in *MODEL*^{ing} are "**Component**", "**Model**" and "**Terminal**". A "**Component**" instance refers directly to the real process component that is to be modelled and contains general attributes like the name of the real component, the process domain and type as well as the reference to different models of the same process. The "**Model**" is an abstraction of the real process component. Several different (i.e., one or more) models could relate to every component, which is indicated by the multiplicity annotation "1+" in the object diagram. Both "**Model**" and "**Component**" have one or more "**Terminals**" of the type "**Input**", "**Output**" or "**Disturbance**".

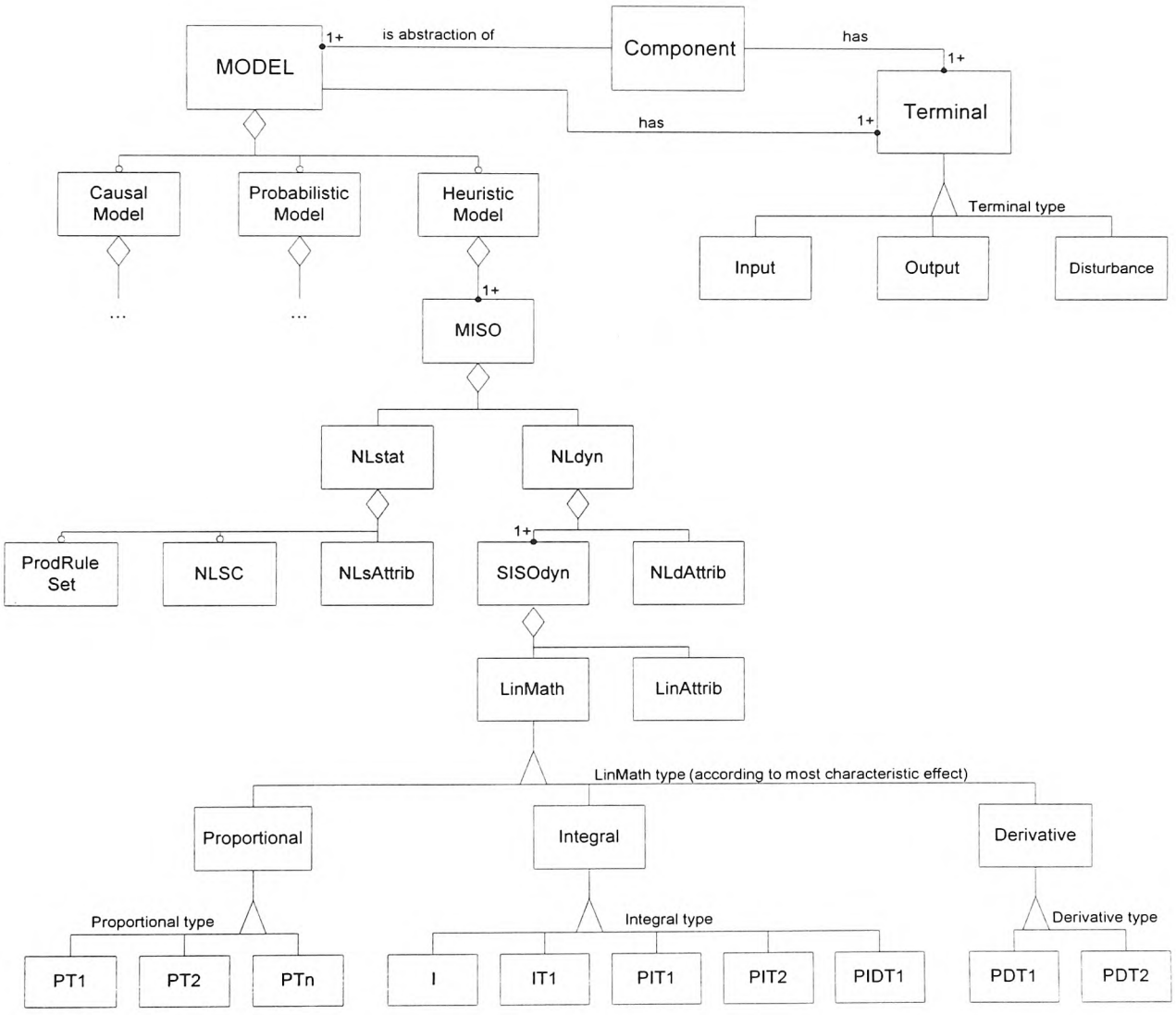


Figure 5-4: The *MODEL*^{ing} Object Model

In general, **Causal-**, **Probabilistic-** and **Heuristic Models** are all ***a kind of*** model (as opposed to "*a part of...*"), which implies an inheritance relation (rather than "aggregation"). The overall model that will result from the extended *MODEL*^{ing} approach, however, will exploit causal, probabilistic and heuristic knowledge, and therefore will typically be an aggregation of the accordingly labelled models. Additionally, case-based knowledge will eventually be exploited through a library based support as part of the other approaches. With this knowledge type being used only as a means of simplifying the causal and heuristic modelling, a separate case-based model will not emerge.

The components of "Causal" and "Probabilistic" model have not been further elaborated here to draw the attention to this work's main focus of interest, the "**Heuristic Model**". A heuristic model - generally multiple input / multiple output, i.e., MIMO - is composed of (or an aggregation of) one or more multiple input / single output (**MISO**) models, with each of the MISO models being related to a different output. In the same way as the MIMO model is built from a collection of MISO parts, MISO in turn is ultimately made up of SISO components. This modular concept, which simplifies the handling of complex multivariable models, is in accordance with the proposed fuzzy hybrid systems view.

Although the emphasis of the modelling approach is clearly on the more comprehensive dynamic modelling, purely static process characteristics can be dealt with separately, too. The reason for this decision is simply that in the envisaged context of application, the available process information may be restricted to static aspects. Therefore, both nonlinear dynamic ("**NLdyn**") and nonlinear static ("**NLstat**") models are parts of the MISO object. "NL", the prefix denoting nonlinearity, indicates that systems are generally expected to have overall a nonlinear behaviour. Linear situations can be considered as special cases within this concept.

In *MODEL*^{ing}, a nonlinear static model ("**NLstat**") can be composed of conventional nonlinear static characteristics ("**NLSC**"), a set of production rules ("**ProdRuleSet**"), and a list of nonlinearity attributes ("**NLsAttrib**"). Whereas the former two objects address well known standard representations of static behaviour (input - output relationships in the form of look-up tables and crisp or fuzzy rules in the form "IF *condition* THEN *consequence*", respectively), the latter type, "**NLsAttrib**" requires some explanation because it is part of a new concept that is proposed in this work:
Complex nonlinear static characteristics could include a variety of nonlinear 'effects' such as discontinuity, intermediate dead zones and hysteresis in a single input - output relationship. Furthermore, it is possible that the effects of such nonlinear behaviour are broadly known in a modelling situation, yet the data to specify "NLSC" cannot be provided. In order to address these

points, the concept of modelling static characteristics by combining applicable attributes was devised. The class "**NLsAttrib**", which has a list of selected linguistic attributes that describe typical features of static characteristics shapes, is the basis for this approach. In the instances of this class, which are created during modelling, these attributes are set as 'TRUE' (i.e., applicable) or 'FALSE' (i.e., inapplicable), according to the knowledge provided in the knowledge acquisition procedure. Further details of this proposal and its importance for the overall goal of this work are given in [108] and section 5.4.7 of this Chapter.

The other part of the MISO object, the nonlinear *dynamic* model ("**NLdyn**"), features also an attributes based representation as one of its parts: "**NLdAttrib**" is the equivalent of "**NLsAttrib**". Hence, the object "**NLdAttrib**" has linguistic attributes that characterise the nonlinear *dynamic* behaviour of the process. Its concept and properties are the same as those of the statics attributes class. The application of "**NLdAttrib**" is further clarified in section 5.4.6.

"**NLdyn**" is, apart from "**NLdAttrib**", built from one or more linear, dynamic single input / single output models ("**SISOdyn**"). One or more of the input parameters (or here 'influence' parameters - cf. section 5.4.3) to the overall model are responsible for the decision as to which one of these linear SISO models is applicable in a particular situation ('fuzTF'-concept).

"**SISOdyn**" in turn is an aggregate of the components "**LinAttrib**", which is a third list of attributes, describing the linear dynamic behaviour of the system, and "**LinMath**". "**LinMath**" stands for linear mathematical models in the form of transfer functions and is subsplit (inheritance relation) into the types "**Proportional**", "**Integral**" and "**Derivative**". The latter three classes serve the purpose of categorising ten standard transfer function structures that are considered in the *MODEL*^{ing} approach. The shorthand that is used for the 'labels' of these standard transfer functions is very simple: 'P' stands for proportional action, 'I' for integral action, and 'D' for derivative action. The 'T' marks a time constant (i.e., indicator for 'lag') and the following number represents the order of the lag. 'PIDT1', for example, means. Transfer function with proportional, integral and derivative action as well as first order lag. Accordingly, 'PTn' stands for a proportional transfer function with n-th order lag.

For the *MODEL*^{ing} approach, the standard transfer functions are categorised according to their most obvious effect in the step response. Transfer functions with "ordinary" proportional step responses (steady state) are of "**Proportional**" type (**PT1**, **PT2**, **PTn**), all those step responses that are steadily increasing (no steady state) are of "**Integral**" type (**I**, **IT1**, **PIT1**, **PIT2**, **PIDT1**) and those that have a steady state but additionally typical derivative effects, like initial impulse-like reaction before

the steady state is slowly approached or non-minimal phase behaviour, are of "**Derivative**" type (**PDT1**, and **PDT2**).

Simple cases of heuristically derived models like linear single input / single output (SISO) are therefore included as sub-sets within this structure that hosts nonlinear multiple input / multiple output cases.

The object structure underlying the "Probabilistic Model", which is not part of the focus of interest in this work and therefore not shown in Figure 5-4, would look very much the same as the "Heuristic Model" structure, except from the absence of any "Attributes" classes (i.e., NLsAttrib, NLdAttrib, LinAttrib). The structure that further defines the "Causal Model", on the other hand, would look significantly different, mainly with classes for algebraic, differential and difference equations to handle different sorts of physical relationships and balance equations. Additionally, the "Causal Model" would have a component class to incorporate causal relationships linguistically. In comparison with the object structures of the "Probabilistic" and "Causal" model, the "Heuristic Model" structure is the most complex one, which is due to the large spectrum of possible information.

All the specific information that is aimed to be acquired through the new approach should normally be integrated in form of attributes of the classes in more refined versions of the object model. However, since the graphical representation of the object model would become difficult to handle, it was decided simply to list the envisaged information. This list is ordered according to the key classes that the information (i.e., attributes or "slots") belongs to and is shown in Appendix A5. Since the attributes should be largely self-explanatory, they are not further discussed here. Many of these attributes are, however, mentioned in the following explanation of the actual knowledge acquisition procedure. Particularly the linguistic attributes of "**NLsAttrib**" are considered in detail at a later stage.

5.4 Analysis And Design Of The Dynamic Model (State Diagram)

It is important to note again that 'design' refers here to the design of the knowledge engineering **methodology**, the approach as such rather than the design of a program, which will mainly serve the purpose of proving the applicability and validity of the approach. Illustrating the design of the knowledge acquisition sequence, the Dynamic Model, which is documented in this section, is obviously one of the particularly important parts of this thesis.

This work is based on the central idea that industrial practitioners **do** have a substantially useful, yet so far unexploited, kind and amount of knowledge which is appropriate at least for general modelling steps. In order to tap this knowledge, the area engineers, who are generally unaware of how to apply the information for modelling purposes, must be guided through the procedure. A simple 'toolbox approach' is therefore inappropriate. For an industrial environment, it is particularly important that the modelling approach is self-explanatory, as the practitioner cannot afford a cumbersome learning process, having to consult a help-system or even a handbook. The key to self-explanatory guidance lies therefore in a carefully structured, stepwise modelling sequence which is partnered up by likewise carefully designed graphical user interfaces (GUI).

The principal design aspects are:

- the required broadness of the approach that enables the exploitation of different kinds of usable information available from process engineers
- the consideration of possible user responses to requested inputs and their further effect on the modelling sequence
- the consideration of the practitioner's point of view - this means in particular that information should be requested in terms of behavioural process effects
- to maintain flexibility and the user's control despite the guidance

All these aspects were considered in the development of the modelling sequence. Illustrating this sequential flow of control and therefore the overall knowledge engineering methodology, the OMT-state diagram of the *MODEL*^{ing} approach is the blueprint for an implementation. Due to its special relevance, the dynamic OMT model was developed in depth on several layers (Figure 5-5 to Figure 5-15).

In order to keep the user largely in control of the modelling process, shortcut facilities to skip individual modelling steps as well as bigger sections of the sequence are available throughout the approach. Likewise, the modelling sequence can be exited altogether virtually at any time. Thus, the user is free to 'ignore' major parts of the sequence and to apply the tools and sub-sections within the approach almost as freely as if they were selected from a simple pull-down menu. By this means, the approach remains flexible despite the guidance.

5.4.1 The Global View Of The *MODEL*^{ing} State Diagram

The top-level state diagram in Figure 5-5 represents the global view at the sequential arrangement of the main modules within the *MODEL*^{ing} approach. Starting from the Main Menu, the sequence commences with the isolation and specification of the component to be modelled. With (possibly partial) completion of this step, the flow of control moves on to the causal knowledge acquisition module, where, in general, the user is requested to provide all known physical equations and relationships among the variables of the system. Parameters that are not exactly known can be specified in terms of value ranges, orders of magnitude or their signs. The module for causal modelling has not yet been further developed.

In the following module for heuristic knowledge acquisition, the user is requested to specify behavioural process information on the basis of experiences from working with the respective 'real world' process. Due to the emphasis of this work on the 'heuristic' module, it is shown with its main sub-sequences - including a recurring loop of MISO modelling, which is repeated according to the number of outputs. Effectively, all modelling of multiple input / multiple output processes is therefore without any loss in generality broken down into several multivariable processes with just one output each. This concept takes the systems view according to the fuzTF approach (Chapter 4) into consideration. MISO modelling in turn is split into the sections Nonlinear Dynamics and Nonlinear Statics modelling which are further detailed in the diagrams of Figure 5-7 to Figure 5-15.

After completing the heuristics knowledge acquisition module, the flow of control moves on to the modelling on the basis of statistical process data, exploiting the probabilistic knowledge type. Like the causal modelling module, this section has not yet been further developed.

Finally, the results in control engineering terms that have been inferred by the *MODEL*^{ing} system from the provided user knowledge are displayed.

After returning to the Main Menu, which is not only possible from the final step (the display of results) but from any point within the modelling sequence, the user can start modelling a new component, browse and edit previously modelled components, delete components, display modelling results of existing components, generate simulation code for a selected component and quit *MODEL*^{ing} altogether.

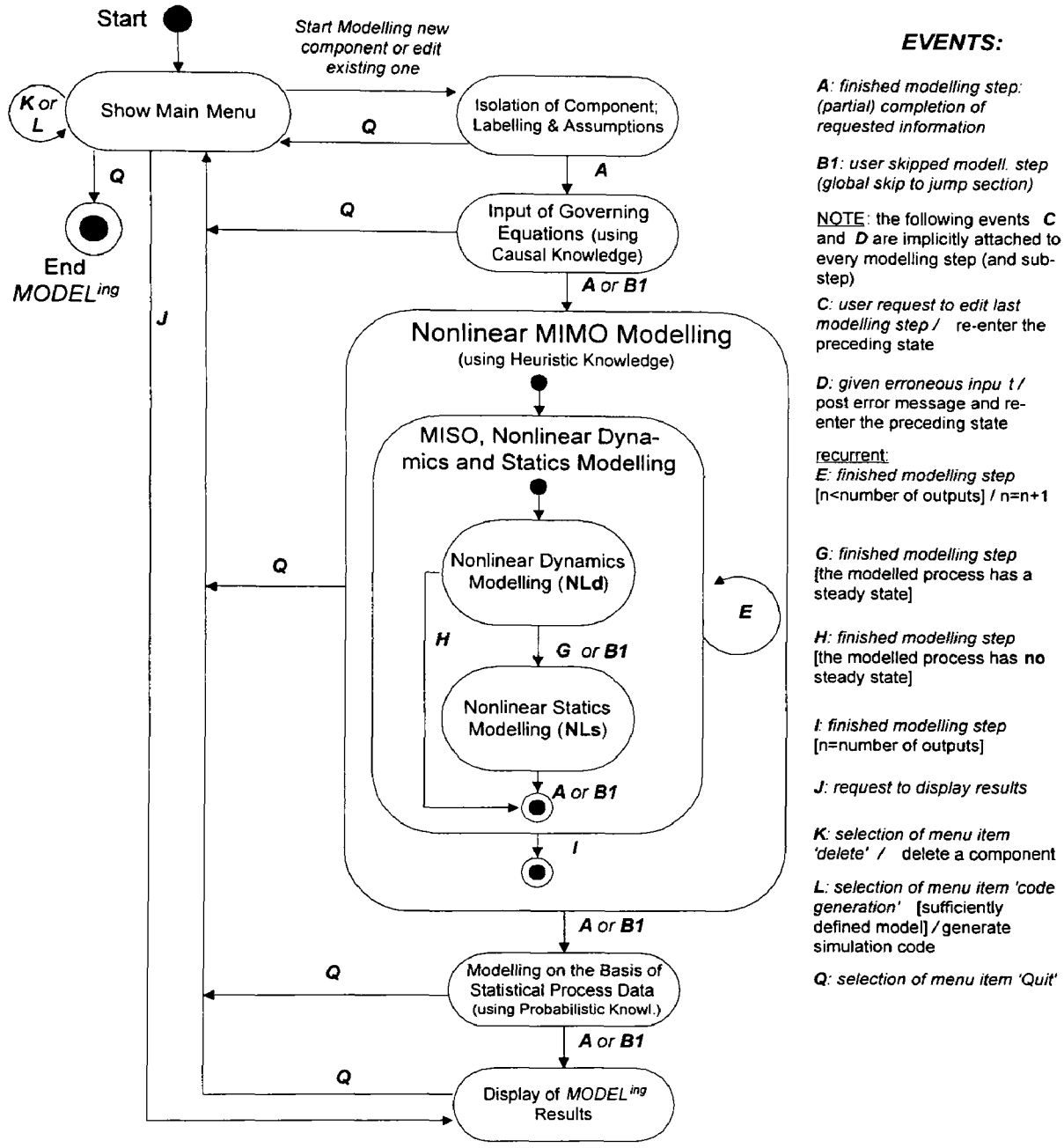


Figure 5-5: Global View Of The *MODEL*^{ing} State Diagram

The design considerations that led to the decision on the sequence of the main sections within *MODEL*^{ing} :

The first type of knowledge to be exploited in the overall *MODEL*^{ing} sequence is the causal knowledge. Causal knowledge is particularly useful for modelling, simulation and the direct translation into a controller design. Virtually all traditional modelling approaches are therefore based

on this knowledge. Since this theoretical kind of process knowledge, which is based on physical balance equations, is normally not readily available for the envisaged *MODEL*^{ing} user group (the practitioners), the causal approach should mainly complement the important heuristic modelling. Previous research in "intelligent modelling" has focused on this area. Particularly noteworthy is the work carried out at Sheffield University between 1987 and 1993 (e.g. [36, 40]). Still though, this field shows scope for further work as the ongoing research by Li et al. [32, 33] shows. Please refer also to the chapter on 'Further Work' in this thesis for suggestions of promising future extensions to *MODEL*^{ing} in this field.

The main focus of the *MODEL*^{ing} approach is put on the next section, the systematic acquisition and evaluation of heuristic knowledge, which has not yet found appropriate attention in the research community. This type of knowledge is normally less attractive from a system scientific point of view (therefore put at the second place in the overall sequence) because the resulting models are often difficult to analyse and usually less precise. On the other hand, heuristically derived models can give insight to the behaviour of processes that cannot be modelled on the basis of balance equations, due to their complexity and possible nonlinear side-effects. Also, the quality of heuristic models can be very satisfying. Due to its salient importance for this work, the heuristic modelling sequence, which exploits the rich practical experience of the area engineer, is exploded to some more detail on this top-level state diagram.

The third knowledge type to be addressed is the probabilistic knowledge which is in the context of industrial process modelling mainly assumed to take the form of statistical quality assurance records. Although this kind of data bears usually little information on system dynamics (thus, third position in the sequence), it could at least be valuable for establishing the main static relationships. Although, of course, many statistical data evaluation approaches exist, the problem has not yet been addressed in this particular context. Similarly to the causal knowledge type, the envisaged further extensions to *MODEL*^{ing} are detailed in the chapter on 'Further Work'.

As was mentioned in the documentation of the object model, the use of case-based knowledge is not explicitly featured in a separate module within the *MODEL*^{ing} concept. Instead, it is suggested to add a library-based approach that runs as a supportive tool concurrently to the 'causal' and 'heuristic' modules, providing on request information in the form of typical answers given during previous modelling sessions for similar types of processes. This supportive library will therefore allow for analogous reasoning within the approach.

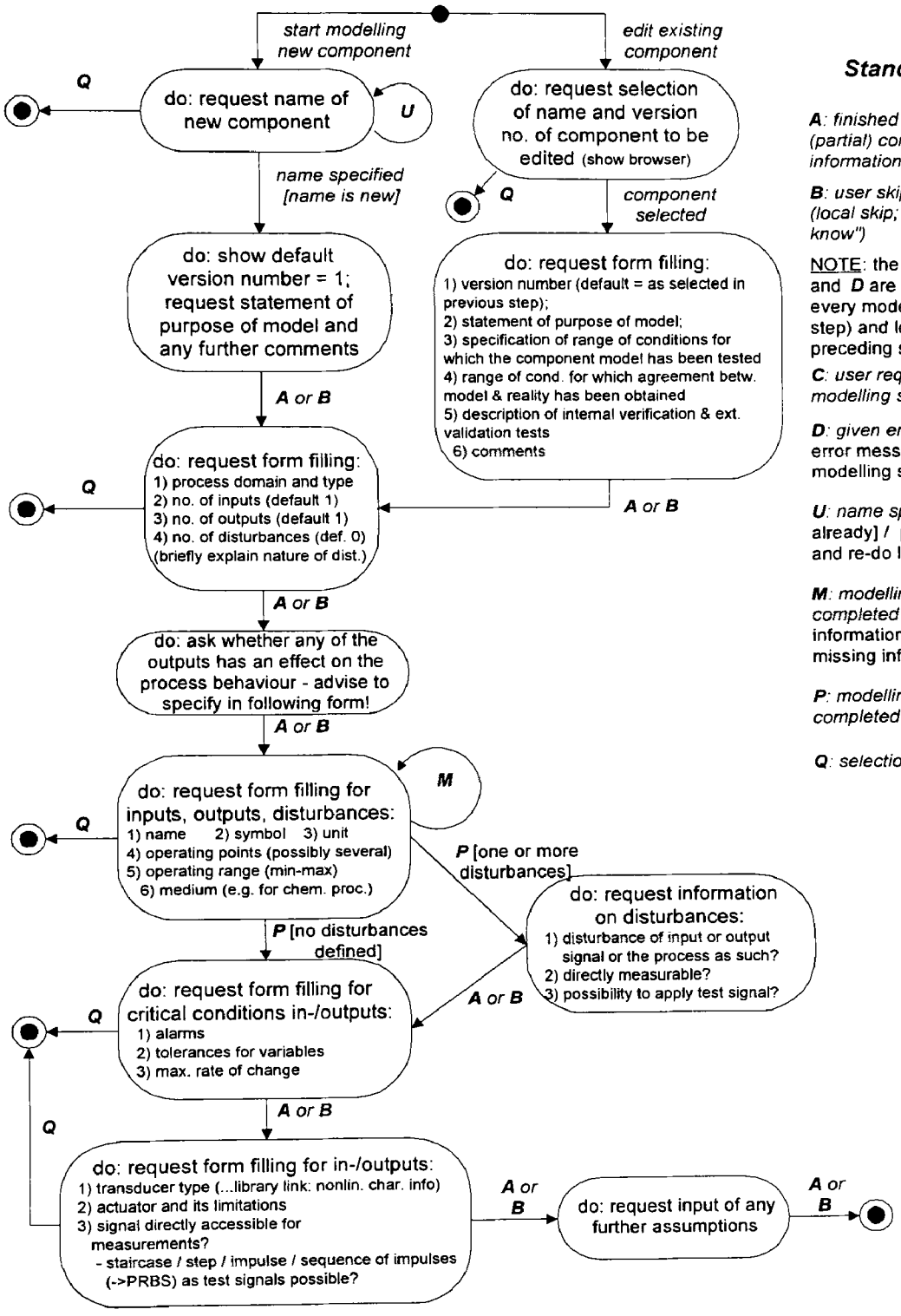


Figure 5-6: Sub-Sequence Of *MODEL*^{ing} : Isolation Of Component

5.4.2 Sub-Sequence Of The *MODEL*^{ing} Approach: Isolation Of Component

One hierarchical level below the global view, this sub-sequence is shown in Figure 5-6.

Depending on whether the user wants to start modelling a new component or edit an existing one, the initial steps in this sub-sequence differ:

In the case of a new component, its name is requested along with information on the purpose of the model. The name must be unique, which is automatically checked. Further optional comments can be added.

To edit an existing component, the user selects the name and version number of the appropriate model from a browser, which supports the stepwise search for models of
a) a particular process domain (1st step) and
b) within that domain the search for models of a particular type (2nd step).
Such a stepwise browsing facility becomes more and more important the bigger the library of previously modelled components grows.

With the help of a form filling approach, he/she can choose between updating the selected version of the considered component and creating a new version on the basis of the old one. Apart from other editing facilities, however, the possibility of documenting validation and verification efforts according to the recommendations of the Society for Computer Simulation, SCS, [118, 123] is of particular importance.

Afterwards, the sequences for 'new component' and 'editing' merge again for the request to specify the numbers of inputs, outputs and disturbances as well as the process domain (e.g. hydraulics) and the process type (e.g. pump). These latter two pieces of information facilitate the above discussed browser support.

To make sure that the user does not forget to specify output parameters as additional inputs in case they influence the process behaviour, this question is explicitly asked, before more details on the process parameters are requested as items in a form: the name of the input, output and disturbance parameters, their physical symbols and units, typical operating points and their operating range. The specification of the medium related to the parameters is mainly of interest in process industry. For example, the medium of the input parameter 'flow' could be 'H_2SO_4'. A checking mechanism ensures that at least the mandatory information (name, symbol, unit, operating point) is provided for each parameter before the flow of control moves on to the next state.

In the case that one or more known disturbances exist, further details that are especially of importance for a possible later experiment design are requested. Afterwards, any critical conditions of process parameters are specified.

The last major state in this sub-sequence allows the input of transducer and actuator types, for which a library based extension is envisaged (see 'Further Work'). Additionally, valuable information for the identification experiment design with respect to allowable signal types can, and should, be entered.

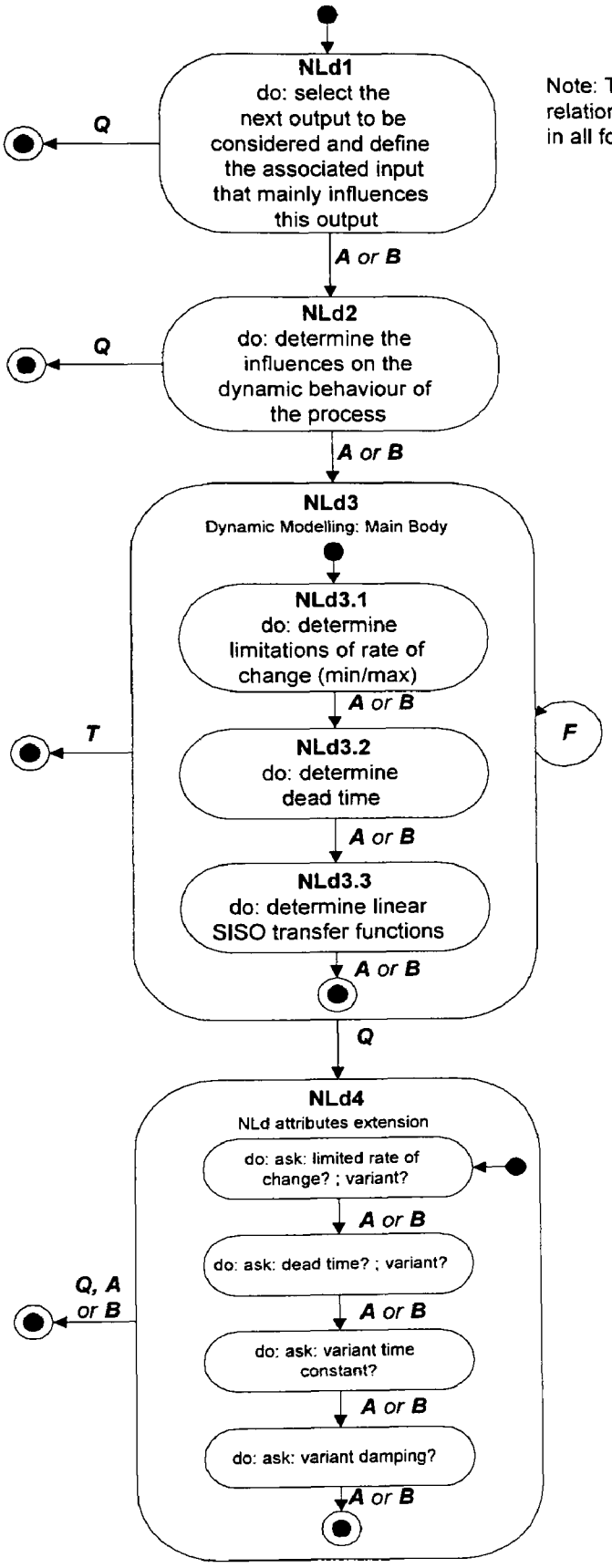
Finally, a facility to specify any further assumptions that have not been covered in the preceding sequence is given.

5.4.3 Sub-Sequence Of MISO Modelling Within The Heuristic Knowledge Acquisition Module: Nonlinear Dynamics Modelling (NLd)

Nonlinear Dynamics Modelling as a sub-state of the heuristic knowledge acquisition module, which is in the global view already split into several sub-sequences, is further detailed in diagram Figure 5-7.

This sequence, which is part of the MISO modelling and therefore repeated according to the number of output variables of the overall component, commences with the selection of the next output variable to be considered in all following steps within the MISO module. Additionally, the associated input variable, which has the most significant effect on the selected output parameter, must be defined. To distinguish between this 'MAIN' input variable and all remaining, less important inputs, the latter are normally only called 'influences' in the following steps. In all these following steps, the currently considered MAIN input - output relation must be clearly indicated to the user. This efficient way of handling the complex issues of multivariability and nonlinearity is not only important for the knowledge acquisition sequence as such but also mirrors the "fuzTF" concept (Chapter 4).

In the next state, the individual influences to be considered for the particular output are selected and typical settings of these parameters are defined. This step is sub-split in section 5.4.4.



Note: The considered main i/o-relation must be clearly indicated in all following modelling steps!

Standard-Events:

***A**: finished modelling step: (partial) completion of requested information*

***B**: user skipped modelling step (interpretation: "don't know")*

NOTE : the following events ***C*** and ***D*** are implicitly attached to every modelling step (and sub-step) and lead to a re-entry of the preceding state

***C**: user request to edit last modelling step*

***D**: given erroneous input / post error message and re-do last modelling step*

***Q**: selection of menu item 'Quit'*

recurrent :
***F**: finished modelling step for the main input-output relation*
[i<number of combinations of distinguished settings of the other inputs, i.e., influences] / i = i+1, next influence vector $!_i$

***T**: finished modelling step for the main input-output relation*
[i=number of combinations of distinguished settings of influences]

Figure 5-7: Sub-Sequence Of MISO: Nonlinear Dynamics Modelling (NLd)

It follows the main body of the dynamic modelling sequence, which is repeatedly accessed in a recurring loop. The concept applied here is to split the MISO modelling into repetitive SISO modelling. Assuming that the above determined MAIN input has always the dominant effect on the currently considered output, the additional influences are considered as side-conditions. Therefore, the SISO dynamic modelling refers always to the MAIN input-output relationship and is repeated as often as there are further combinations of distinguished influence parameter settings (= side-conditions) that the user can or wants to consider. Thus, each time the "Dynamic Modelling: Main Body" is accessed, a fixed set of influence parameter settings must be defined.

"Dynamic Modelling: Main Body" is sub-split into the acquisition of information on limited rates of change and dead time as well as the modelling step that should yield linear SISO transfer functions. The dynamic modelling sequence is further detailed in 5.4.5 and 5.4.6.

With the completion of the sequence in the main body of dynamic modelling, the nonlinear dynamics modelling (NLd) is normally complete, too. *Only* in the case that the "Main Body" cannot be handled appropriately, due to lacking process information, it is exited and a separate sequence, which aims at extracting some attributes of nonlinear dynamic behaviour, is accessed.

5.4.4 Sub-Sequence Of NLd: Determination Of The Influences On The Dynamic Behaviour Of The Process

The purpose of this sequence shown in Figure 5-8 is the selection of relevant influence parameters (apart from the MAIN input) with respect to the *dynamic* behaviour of the considered output among the previously specified overall input parameters to the modelled component. Obviously it is assumed here that the general situation of a multiple input / multiple output process prevails, where by far not all process inputs are necessarily relevant to all outputs.

In addition to selecting the applicable influences among previously specified parameters, however, further influences can be defined, if they have been forgotten before. Since particularly the influence of time as well as the direction of input change (i.e., sign(du/dt)) are easily overlooked as potential influences, the user is explicitly reminded of these possibilities, unless he/she is absolutely sure that the dynamic i/o-relationship (= MAIN input / output relation) is invariant under all circumstances.

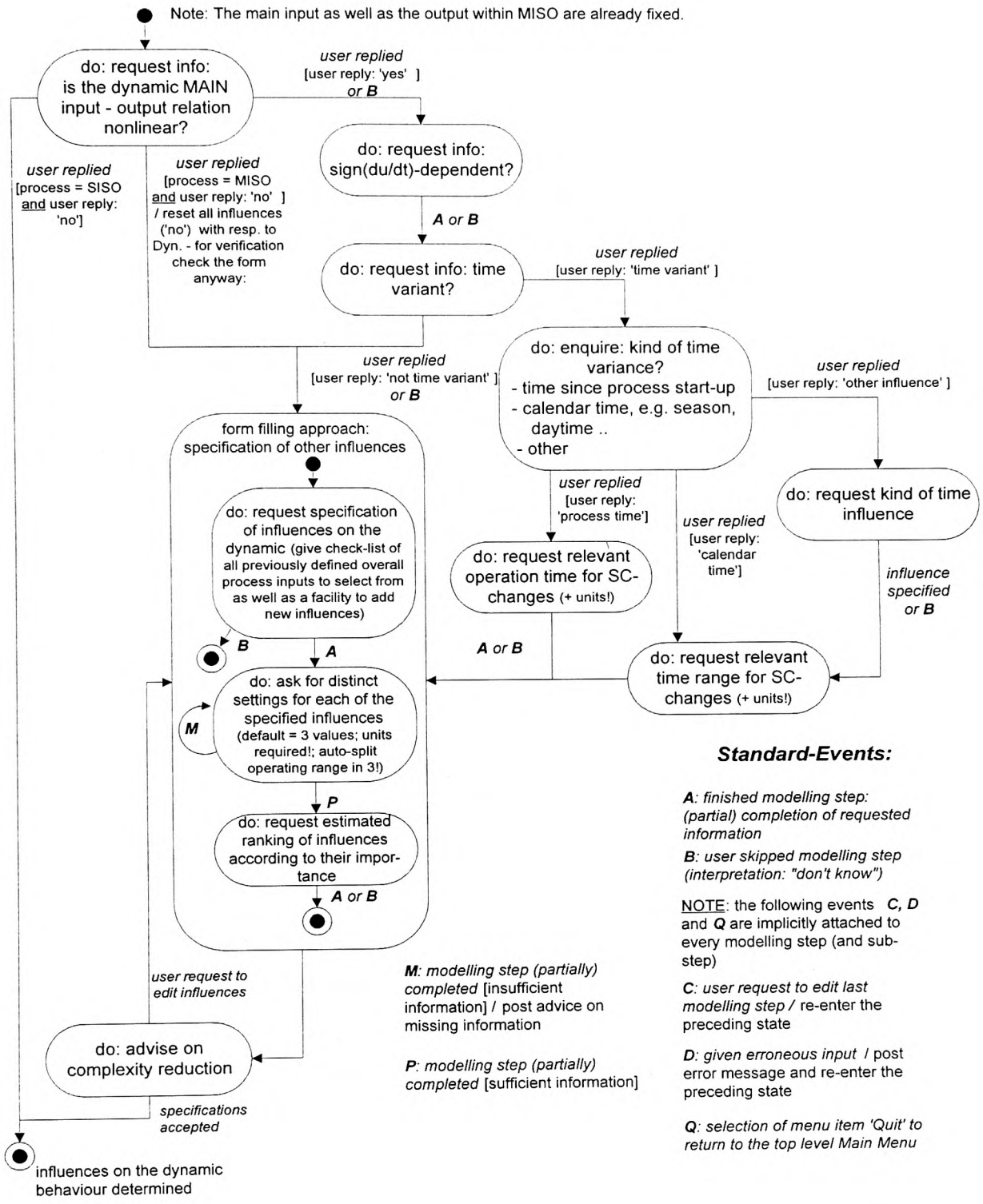


Figure 5-8: Sub-Sequence Of NLd: Determination Of Influences On Dynamics

After the selection or specification of influences, the user is requested to define distinct settings of these parameters within their operating range which lead to a significantly different main i/o-

relationship. Advice is given in case that too many parameters and settings are specified (high complexity).

Additionally, the user should give an estimated 'ranking' among the influences, according to their importance.

5.4.5 Sub-Sequence Of NLd: Main Body Of The Dynamic Modelling Sequence

The modelling sequence illustrated in Figure 5-9 shows in greater detail the above mentioned dynamic modelling sequence.

Initially, two particular nonlinear dynamic effects are addressed: information regarding dead times as well as the presence of limitations in the rate of change for both in- and decrease of any process parameter is requested. Ideally, numerical information with respect to these effects should be specified, too. By isolating these effects first, the following modelling sequence can be substantially simplified.

Afterwards, the separately framed linear single input / single output (SISO) modelling sequence is accessed. This SISO sequence is based on the selection of the step response shape that resembles most closely the real behaviour of the process component. A decision support function is provided in case that the user is unable to make a direct choice. According to the selection of the step response type, some specific pieces of numerical information are requested, which are used for the calculation of approximate transfer function parameters using basic process identification approaches. Only in case that either the decision support fails to enable the user to choose between the step responses or none of the requested characteristic step response parameters can be provided, an alternative, simplified approach on the basis of a first order proportional model with dead time is taken.

The framed linear SISO modelling within the main body of the dynamic modelling sequence, which is completed by a display of intermediate results, is further detailed in the following section.

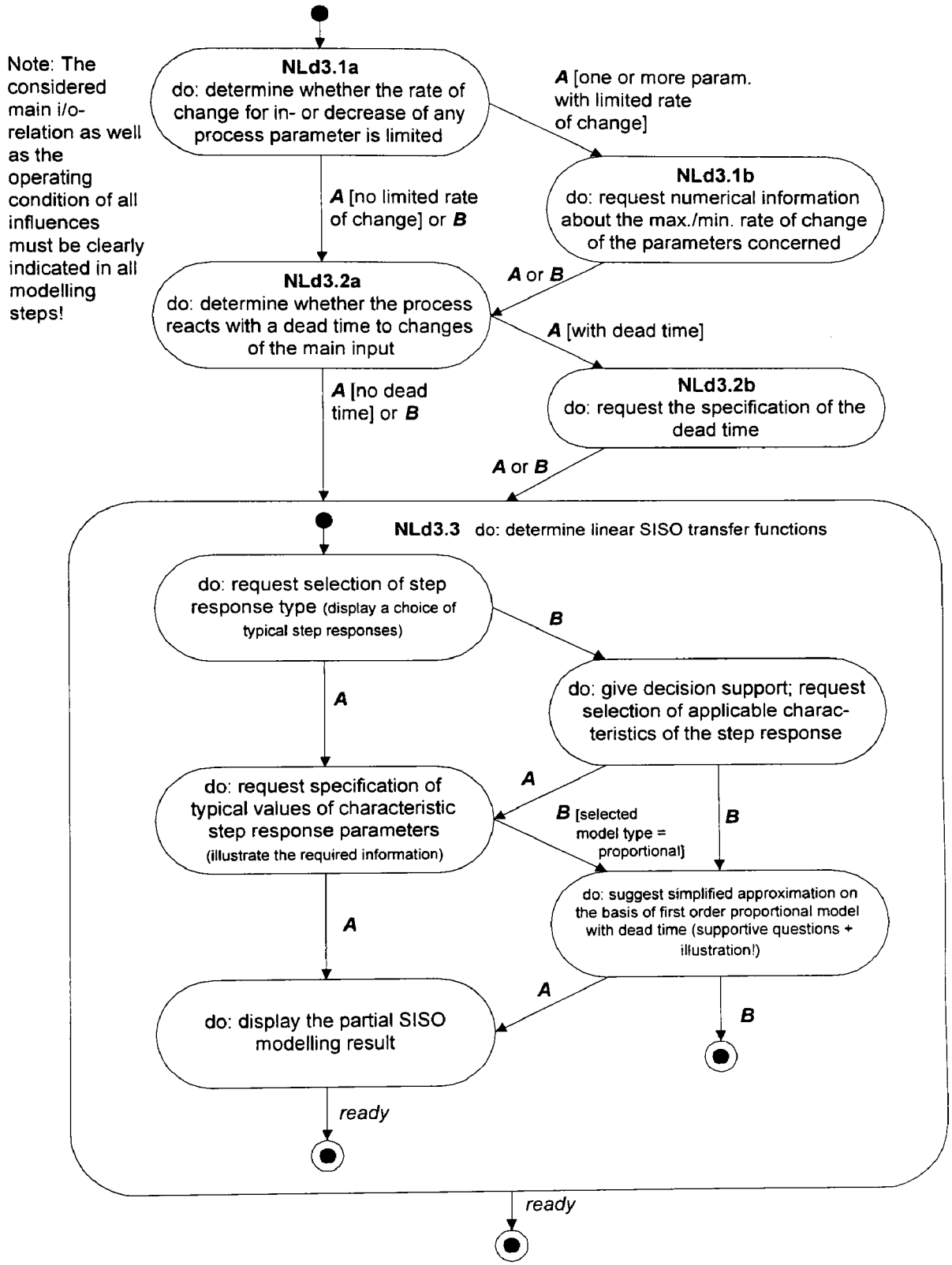


Figure 5-9: Sub-Sequence Of NLd: Main Body Of Dynamic Modelling

5.4.6 Sub-Sequence Within The Main Body Of The Dynamic Modelling Sequence: Determination Of Linear SISO Transfer Functions

Further to the preceding description of the general concept of linear SISO modelling based on the selection of the step response shape, this section gives more detailed considerations of this design step and Figure 5-10 focuses in particular on the design of the decision support system.

Graphical representations of typical step responses are particularly useful as they allow for accessing the process expert's understanding of the temporal process behaviour. Step responses are ideal for this purpose, since they are straightforward to understand, easy to reproduce and because they often relate closely to process behaviours which the area engineer is familiar with through plant chart recorder records. With the decision for the step response graph that matches the real process behaviour best, the area engineer gives therefore indirectly a very detailed piece of information without being bothered with theoretical details.

The displayed choice of step responses, amongst which the user can select (Figure 5-10), corresponds to the ten sub-classes of 'Math', which are assorted in the intermediate classes 'Proportional', 'Integral' and 'Derivative' according to their salient characteristic, as was discussed in 5.3. Since the shape of the response can have very distinct features for both 'PDT1' and 'PDT2' transfer functions, depending on the parameter settings, these two types must be represented by two typical step responses each. Thus, the overall choice of typical step responses increases to twelve.

If the user is able to select one of the step responses directly, the flow of control moves directly on to the request to specify typical values of characteristic step response parameters, as was mentioned in the previous section. To clarify the required information, an expanded figure of the selected step response, which clearly indicates the characteristic values, must be shown.

The set of implicit answers to be given by the selection of a step response, however, could exceed the abilities of the area engineer in some situations. Therefore, the direct selection of an appropriate step response is complemented with a decision support facility. Conceptually, the decision support facility is based on a question-answer mode to narrow the choice of possible step responses incrementally. An important aspect in the design of this question-answer mode is the minimisation of the number of required questions, so that firstly, the approach is not too cumbersome for the user and that, secondly, specific questions which are only of interest in particular situations are not generally asked since they could confuse users in other modelling situations.

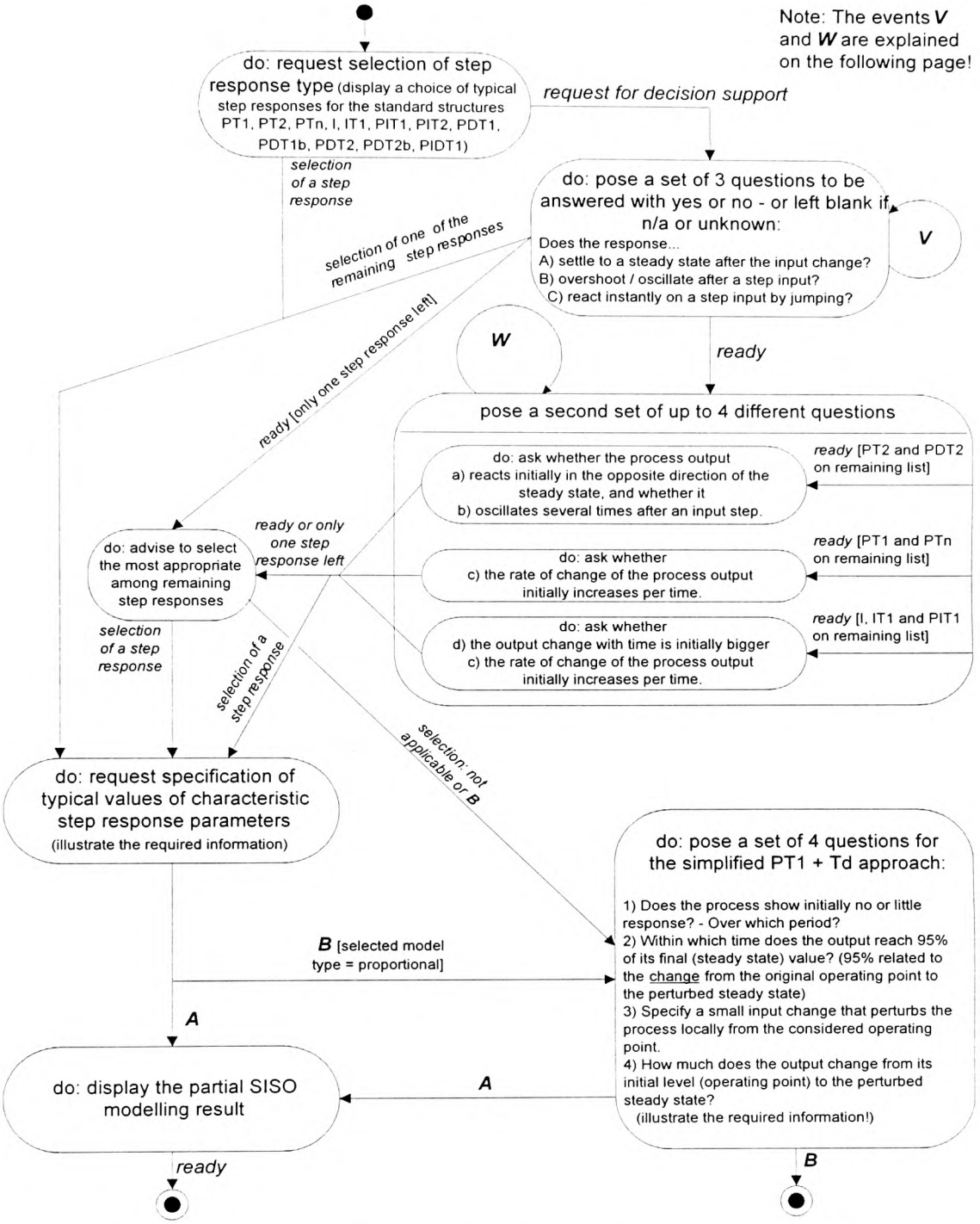


Figure 5-10: Sub-Sequence Of The Dynamic Modelling Main Body: Determination Of Linear SISO Transfer Functions

Annotations to Figure 5-10:

V**: user replies* - six different types of event ***V are distinguished:
***V1**: user replies* [A) Yes, steady state] / remove as inapplicable from the selection list: PIT2, PIDT1, I, IT1, PIT1
***V2**: user replies* [A) No steady state] / remove from selection list: PT1, PT2, PTn, PDT1, PDT2, PDT2b, PDT1b

***V3**: user replies* [B) Yes, overshoot or oscillation] / remove from selection list: I, IT1, PIT1, PT1, PTn, PDT1b
***V4**: user replies* [B) No overshoot or oscillation] / remove from selection list: PT2, PIT2, PDT2, PDT2b, PDT1
***V5**: user replies* [C) Yes, instantly impulse-like] / remove from selection list: I, IT1, PIT1, PIT2, PT1, PT2, PTn, PDT2, PDT2b
***V6**: user replies* [C) Not instantly impulse-like] / remove from selection list: PIDT1, PDT1, PDT1b

W**: user replies* - eight different types of event ***W are distinguished:
***W1**: user replies* [a) Yes, initially opposite] / remove as inapplicable from the selection list: PT1, PT2, PTn, I, IT1, PIT1, PIT2, PDT1, PDT1b, PDT2, PIDT1
***W2**: user replies* [a) Not initially opposite] / remove from selection list: PDT2b
***W3**: user replies* [b) Yes, several oscillations] / remove from selection list: I, IT1, PIT1, PT1, PTn, PDT1, PDT1b, PDT2, PIDT1
***W4**: user replies* [b) Not several oscillations] / remove from selection list: PT2, PIT2
***W5**: user replies* [c) Yes, accelerating change] / remove from selection list: PT1, I, PIT1, PDT1, PDT1b, PDT2, PDT2b, PIDT1
***W6**: user replies* [c) No accelerating change] / remove from selection list: PTn, IT1
***W7**: user replies* [d) Yes, initially faster] / remove from selection list: PTn, I, IT1
***W8**: user replies* [d) Not initially faster] / remove from selection list: PT1, PIT1, PDT1, PDT1b, PIDT1

The formulation of the questions is meant to be as simple as possible. Yet, the area engineer will still have to think carefully about one or the other question during the application of the approach since some distinctions are quite difficult to be put in plain words. All questions are answered by 'yes' or 'no'. Only the three most important standard questions are initially asked:

Does the step-response
A) settle to a steady state after the input change?
B) either overshoot or oscillate after a step input?
C) react instantly on a step input by jumping?

Question A) separates processes with integral action from those without. In question B), processes that either overshoot or oscillate are separated from those that show neither of these characteristics. The idea of combining the two effects in question B) is based on the likelihood that the user will not always be able to tell them apart in a practical situation. Hence, the distinction between overshoot and oscillation is made at a later stage. Question C), finally, aims at separating processes with derivative action and first order lag from those without derivative action or higher order lag.

After answering the first three questions, the choice of possible step responses has ideally gone down to a single one, which should then be picked. In most situations, however, a second set of up to four more specific questions must be provided - possibly even because not all of the initial three

questions have been answered. This second set depends on the remaining response types that must be further distinguished. Previous answers are therefore indirectly considered so that only sensible and important questions for the specific situation are asked:

If 'PT2' and 'PDT2' are among the remaining standard structures, the questions

"Does the process output react initially in the opposite direction of the steady state? "
and
"Does the output oscillate several times after an input step?"

are asked.

The first question should isolate the non-minimal phase type of the two 'PDT2'-responses from the minimal phase one (and, in fact, from all other responses), whereas the second question distinguishes the 'PT2'-response, which is considered to be the preferential structure to represent oscillating proportional processes, from other responses.

If 'PT1' and 'PTn' are among the remaining structures, the question

"Does the rate of change of the process output initially increase per time?"

is asked.

Since 'change per time' refers to the first derivative of the step response graph, the question points directly to the characteristic difference between first and n-th order proportional behaviour (with n > 1).

The latter question is also asked if 'I', 'IT1' and 'PIT1' are among the remaining choice of standard responses, because a positive answer points here at 'IT1'. In order to further distinguish between 'I', which features a constant output change after a step input, and 'PIT1', the question

"Is the output change with time initially bigger?"

is asked.

In case that all of the questions posed in an arbitrary modelling situation are answered with 'yes' or 'no', the decision process will always narrow the original choice of twelve standard responses down to one.

It is important to evaluate all questions immediately after the answer is given (events ***V*** and ***W*** in Figure 5-10), so that the user can keep track of - and learn from - the selection process.

Since the whole approach focuses on the practical, behavioural point of view, some of the decisions are not made in the scientifically exact manner. A 'PTn' transfer function with n > 2, for example, could be oscillating, yet it is sufficient to approximate such cases here always with the 'PT2' structure. Likewise, actual 'PDT2' responses could look very much like 'PTn', in which case the simpler 'PTn' structure would always be applied.

At first sight, the decision support might seem to be a bit trivial or especially geared at educational purposes. A closer look at systematic procedures for problem solving reveals, however, that it is most important to ask the right questions. This applies as much to problem analysis in everyday life as to troubleshooting in industry and to process modelling. Formulating the right questions in order to find the relevant answers becomes all the more difficult the less one knows about the problem domain. An experienced process engineer in industry, for example, who has to analyse and solve a problem which occurred with one of the processes is perfectly able to formulate the problem-specific questions, find out the answers and thereby solve the problem. The same process engineer has probably sufficient process knowledge to answer the questions which are relevant to process modelling for control purposes, but without sufficient modelling knowledge he/she would not be able to ask the relevant questions in the first place.

Also, it could well be that it is not only impossible to make a direct selection of a typical step response in a given situation, but also to answer all required questions in the decision support process. In such a situation, the decision support enables the engineer to enter at least the partial knowledge which is available. Even if the user is not able to have a 'clever guess' among the reduced choice of responses, this partial knowledge is still very valuable since it is accumulated in the "**NLdAttrib**" attributes list (see object diagram), which improves the systematic design of process identification experiments in a separate CACSD module.

If the step response that results from the completed decision support approach is accepted, the flow of control moves on to the request to specify typical values of characteristic step response parameters - the same state that is immediately entered in case a direct selection of the step response is made (see above). If either the user is not happy with the suggestion resulting from the

decision support or he/she cannot specify the required characteristic step response parameters, a simplified alternative approach is taken. This alternative approach, which is only applicable to the approximation of process behaviours without integral action, is based on the standard structure of a dead time followed by a first order proportional transfer function. In order to approximate the real behaviour with this simplified - but astonishingly versatile - standard approach, an input form, requesting some basic step response information, is presented to the user (last state in Figure 5-10).

5.4.7 Sub-Sequence Of MISO Modelling Within The Heuristic Knowledge Acquisition Module: Nonlinear Statics Modelling(NLs)

Following the Nonlinear Dynamics Modelling (NLd), the important Nonlinear Statics Modelling state is already shown in the global view of the *MODEL*^{ing} sequence. In the following, this state is split into its sub-states.

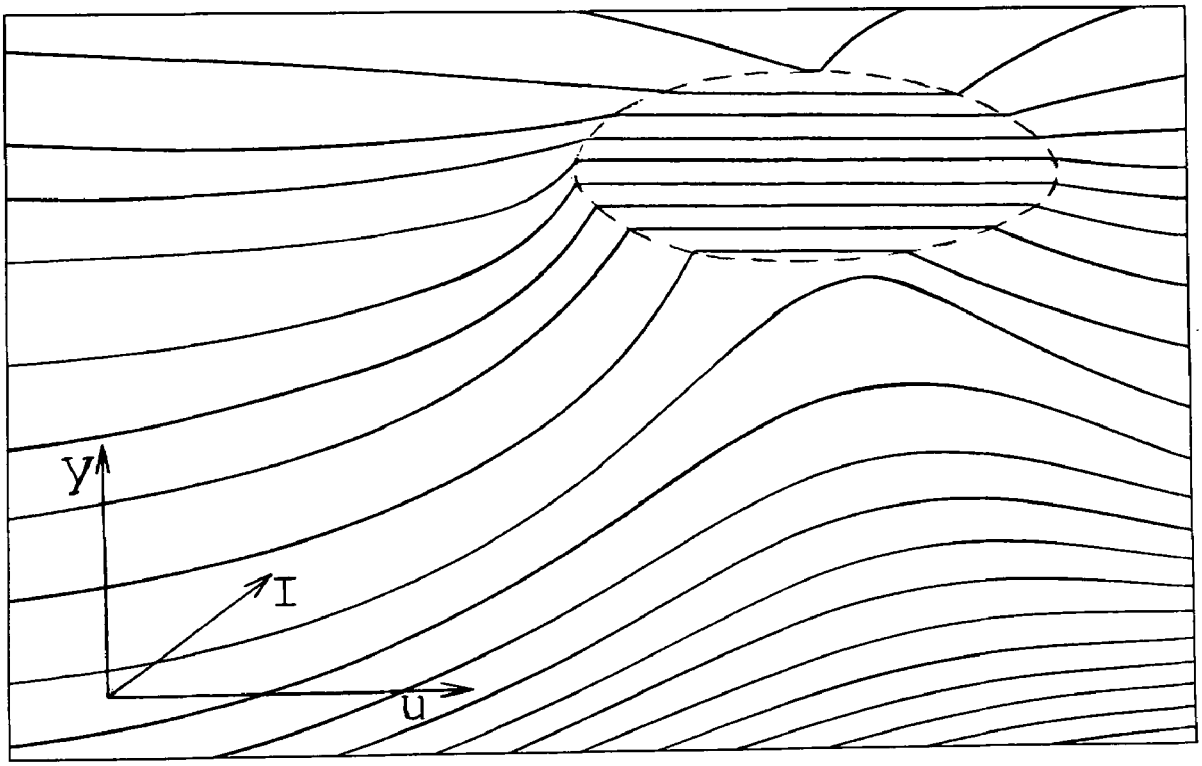


Figure 5-11: Multidimensional Static Characteristics As Collections Of 2-Dimensional Sectional Views

(Parallel 'Cuts' To The MAIN Input-Output Plane U-Y, each at a fixed setting of influence I)

Firstly, however, the reader is reminded of the concept of handling multivariable static characteristics in this work. Similarly to the dynamic modelling, the relation between each output

and its MAIN input is in the centre of interest. This means, that exclusively the two-dimensional relationship between MAIN input and output is modelled - but repeatedly, for the different settings of the additional influence parameters. Thus, the 2-dimensional characteristics can be considered as sectional views of the n-dimensional static characteristic. Similarly to multiple slices through a surface, the collection of 2-dimensional static characteristics defines the complete multivariable characteristic (Figure 5-11).

Figure 5-12 gives an overview of the NLs-approach. After the successful completion of the nonlinear dynamic modelling approach, the NLs-approach will often not be required, which is the reason for requesting explicitly the consent of the user to proceed. In the following state, the dimension (i.e., complexity) of the static characteristic is determined. After this preparation, the actual static modelling commences in the following step, where the user is requested to specify static input-output relations either directly or in form of production rules.

These direct ways of specifying the static relationships are complemented by an interactive approach that is aimed at extracting descriptive attributes for the shape of the static characteristics (cf. considerations on "**NLsAttrib**" in the object diagram). In comparison with the attempt to determine standard types of static characteristics (like backlash, friction or hysteresis), this new modelling approach is better able to cope with combinations of different nonlinear effects in a modular and flexible fashion.

The considered attributes are:

- continuous static characteristic (SC)
- SC with positive and/or negative slope
- SC with upper and/or lower limitation
- maxima, minima in the SC
- SC with intermediate dead zone
- multiple valued / multiple variable[2]
- time variant

...as well as the negation of these attributes.

Beyond the Boolean yes/no-decision as to whether an attribute would be appropriate for the description of the considered static characteristic, the *MODEL*^{ing} approach tries to elicit further, in particular numerical, information from the user.

[2] information on 'multivariable' / 'multiple valued' and time variance is already acquired in the state "NLs2: determine complexity and influences of static characteristic"

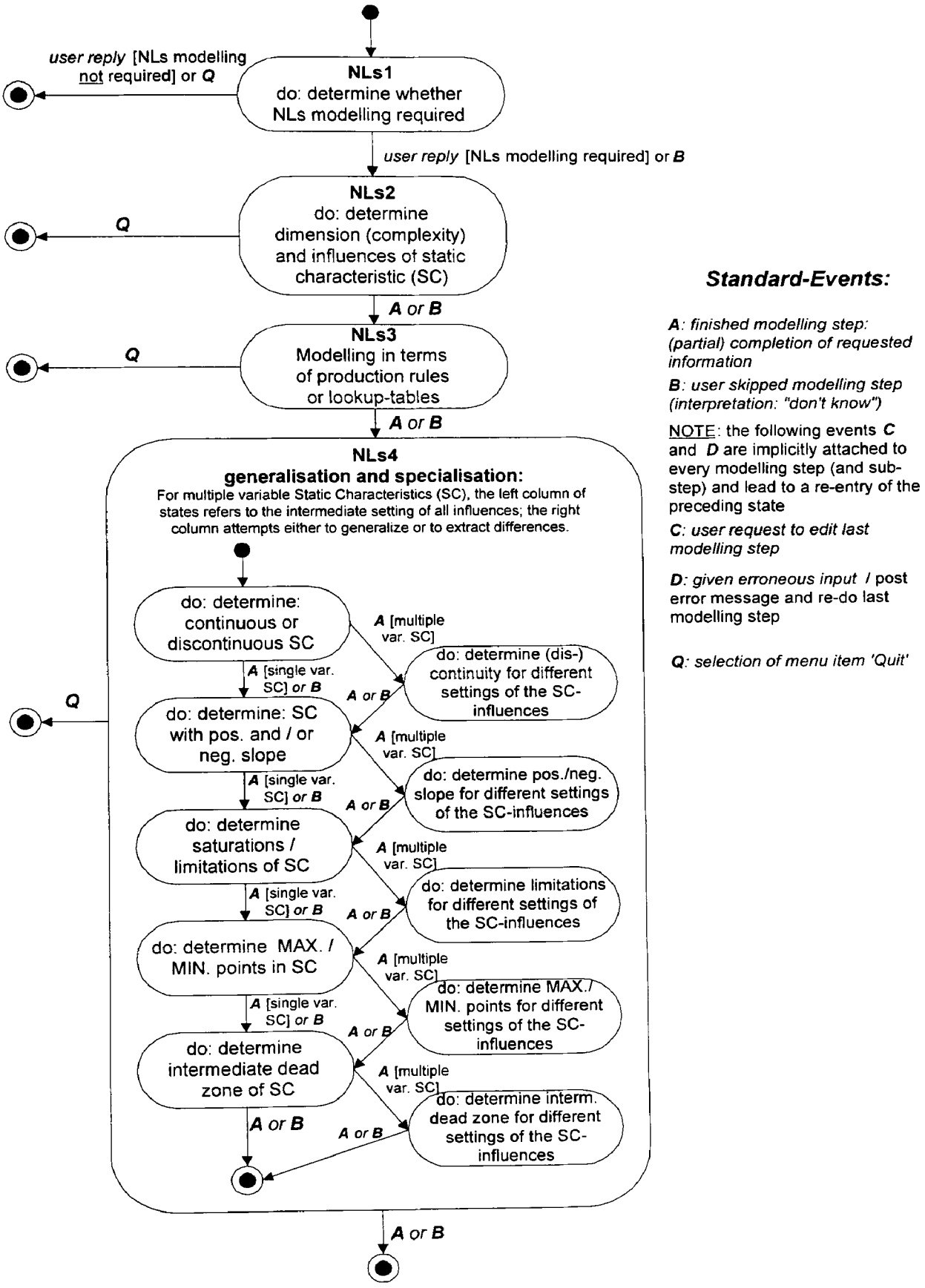


Figure 5-12: Sub-Sequence Of MISO: Nonlinear Statics Modelling (NLs)

This new, attributes based approach is the most abstract level of modelling featured in the proposed knowledge acquisition methodology. Nevertheless, such an attributes list with information on nonlinearities greatly simplifies the design of identification experiments in that it helps either to avoid critical areas of the operation range, or to run specific tests to gain more information about the nonlinearities.

In the simple case of only one input variable without any further influence parameters, the attributes are successively checked (i.e., left column of sub-states within the "generalisation and specialisation" box in Figure 5-12). However, the situation becomes much more complicated whenever additional influences affect the MAIN input-output relationship. To address this situation systematically, the "generalisation and specialisation" approach was developed:

For multivariable static characteristics, the left column of states within the "generalisation and specialisation" box refers to the MAIN input-output relationship at intermediate settings for all additional influence parameters. After the completion of each step in the left column, the approach attempts to generalise the results to as many other settings of the influences as possible. For those influences settings that result in a distinct shape of the static MAIN input-output relationship, the information must be specialised.

The particularity about this approach is the "generalisation" concept, which allows for a significant reduction of the complexity in the knowledge acquisition sequence, while the "specialisation" facility retains the flexibility of the overall approach. Apart from the typical reduction of the workload, the approach protects the user from having to face the whole complexity at once, which is an important motivation factor.

5.4.8 Sub-Sequence Of NLs: Determination Of The Complexity Of The Static Characteristic

The purpose of the sequence in Figure 5-13 is equivalent to the dynamic modelling sequence in Figure 5-8: the selection of parameters other than the MAIN input that are relevant to the shape of the static relation between MAIN input and output.

Since the sequence is largely identical to the dynamic modelling sequence, please refer to section 5.4.4 for a brief description.

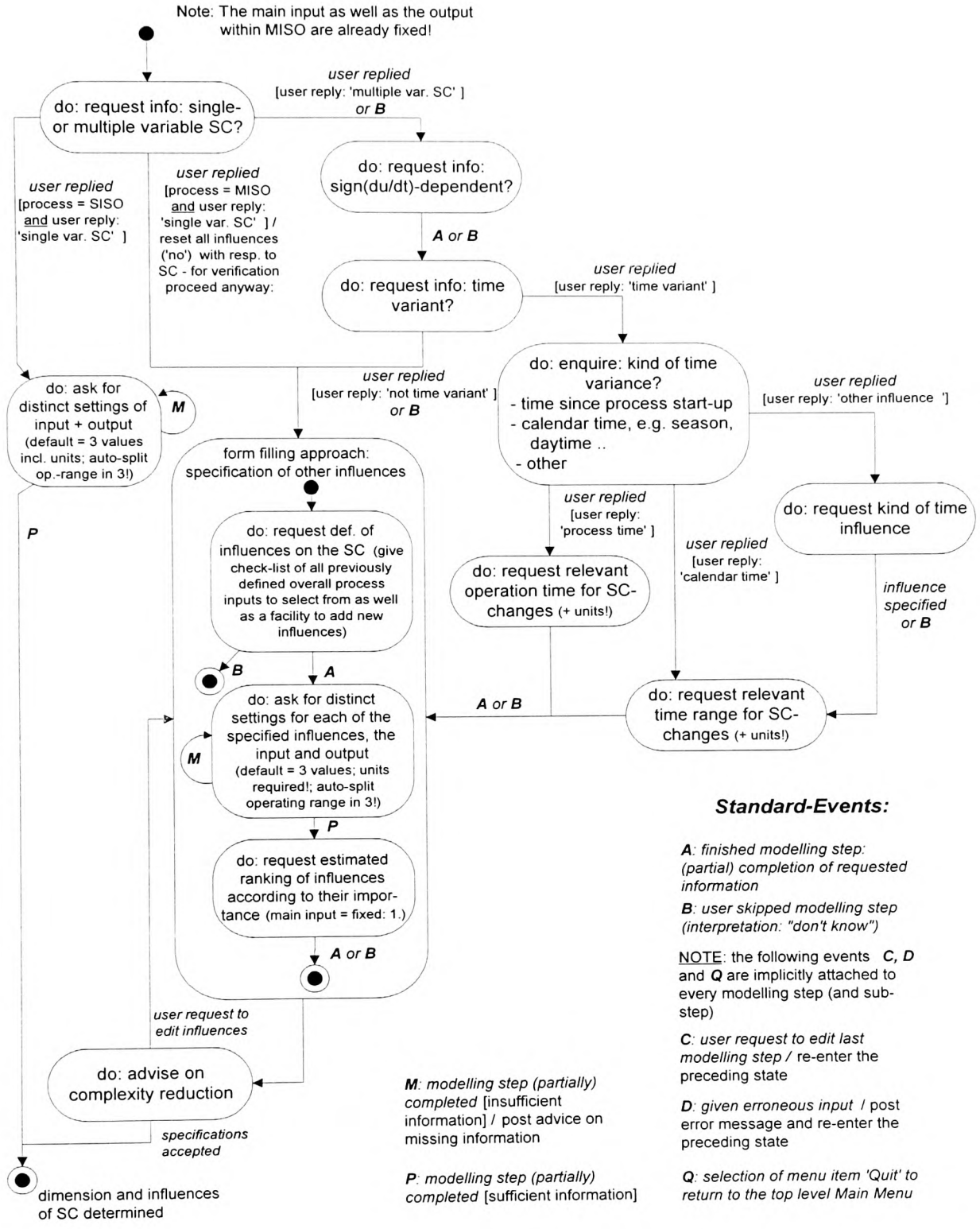
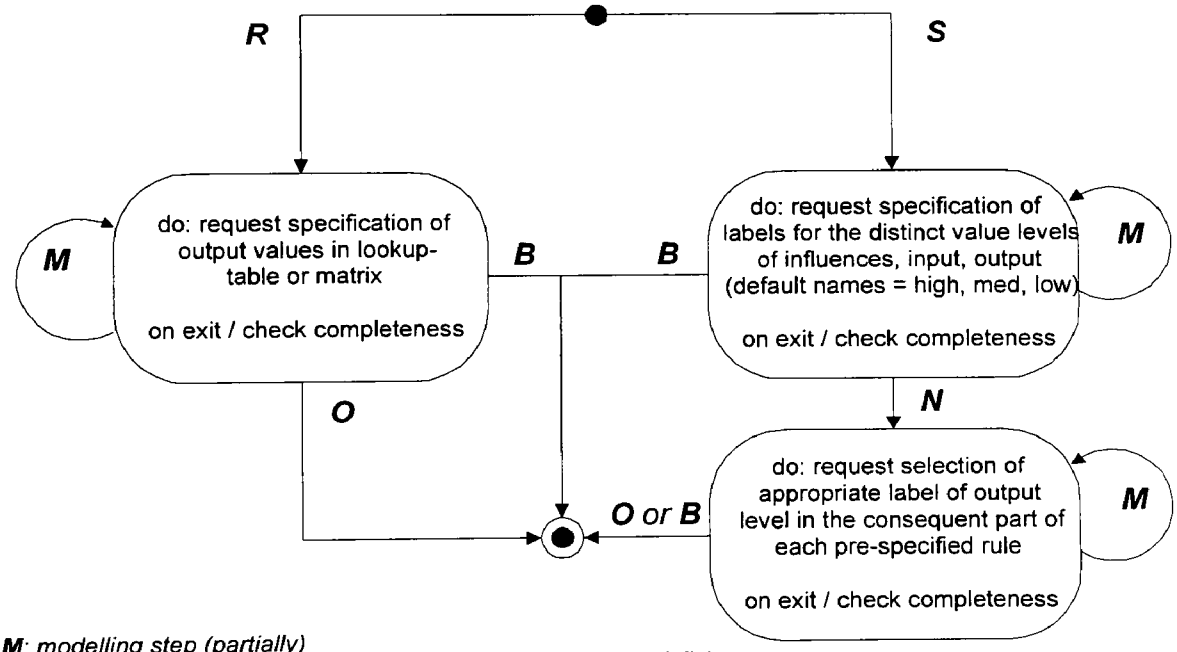


Figure 5-13: Sub-Sequence Of NLs: Determine Complexity Of Static Characteristic

5.4.9 Sub-Sequence Of NLs: Modelling In Terms Of Production Rules Or Lookup-Tables

In Figure 5-14, the flow of control splits right at the beginning, depending on the complexity of the static relationship:

1. If 0 or 1 influence must be considered in addition to the MAIN input, the user is requested to specify the appropriate output levels directly in a table or a matrix for each pre-defined level of the MAIN input or both the MAIN input and the additional influence, respectively.
2. For 2 or more influences in addition to the MAIN input, a rule-based modelling approach is taken. In this case, the user is firstly requested to define linguistic labels like 'high', 'medium', 'low' for the previously defined parameter values of input, influences and output. After the automatic generation of the complete antecedents (IF ...) for all possible rules as well as their consequent parts with only the output labels to be defined, the user is requested to complete the consequents simply by choosing the appropriate output label for each rule (e.g., ...THEN output is 'low').



***M**: modelling step (partially) completed* [insufficient information] /post advice on missing information

***N**: modelling step (partially) completed* [sufficient information] / generate rules: all antecedents complete with fuzzy set labels + consequents with the choice of output fuzzy sets to select from

***O**: modelling step (partially) completed* [sufficient information] / complete the rules with specified THEN part + delete all rules with incomplete consequent

***R**: dimension and influences of SC determined* [0 or 1 influence in addition to the main input to be considered]

***S**: dimension and influences of SC determined* [2 or more influences in addition to the main input to be considered]

***B**: user skipped modelling step (interpretation: "don't know")*

Figure 5-14: Sub-Sequence Of NLs: Rule-Based Modelling, Lookup-Tables

5.4.10 Sub-Sequence Of The Generalisation And Specialisation Approach Within NLs

This static modelling approach aims always at the acquisition of attributes (and possibly data) that refer to the relationship between MAIN input and output at a particular setting of the additional nonlinear influences. By this means, one of the 2-dimensional sectional views of the multidimensional static characteristic is defined (cf. Figure 5-11). The "Generalisation and Specialisation" approach aims then firstly to generalise the provided information for the adjacent influences settings (i.e. the sectional views of parallel planes, Figure 5-11). Only if the user does not agree with this generalisation, the "Specialisation"-part facilitates the definition of different attributes and/or data that refer to the static MAIN input - output relation at other settings of the nonlinear influences. Wherever results can be generalised, the user can save a lot of time and effort through this approach.

The sub-sequence in Figure 5-15 details the exact flow of control that is required to obtain the flexible "Generalisation and Specialisation" approach. For each of the attributes that are individually considered to extract as much information from the user as possible, the sequence consists of four states:

- The enquiry about the applicability of the attribute or several related attributes, possibly supported by a graphical illustration (by default, at intermediate settings for all additional influences).
- The request to specify some data around the particular area of the static characteristic.
- The enquiry as to whether the previous answers apply irrespective of the settings of the influences (i.e., "can the results be generalised?").
- The request to select a new combination of settings of the influences that ought to be considered (i.e., the specialisation approach).

These four states are combined by a circular flow of control, so that this sequence is repeated for different influence settings, unless it is exited. The most important one among the exits from each circular sequence is the one leaving the third state by answering "Yes" ("..., the results can be generalised for the other influences settings").

The aim of this approach is to obtain the "minimum" type of information, a list of attributes, and possibly even some additional numerical information.

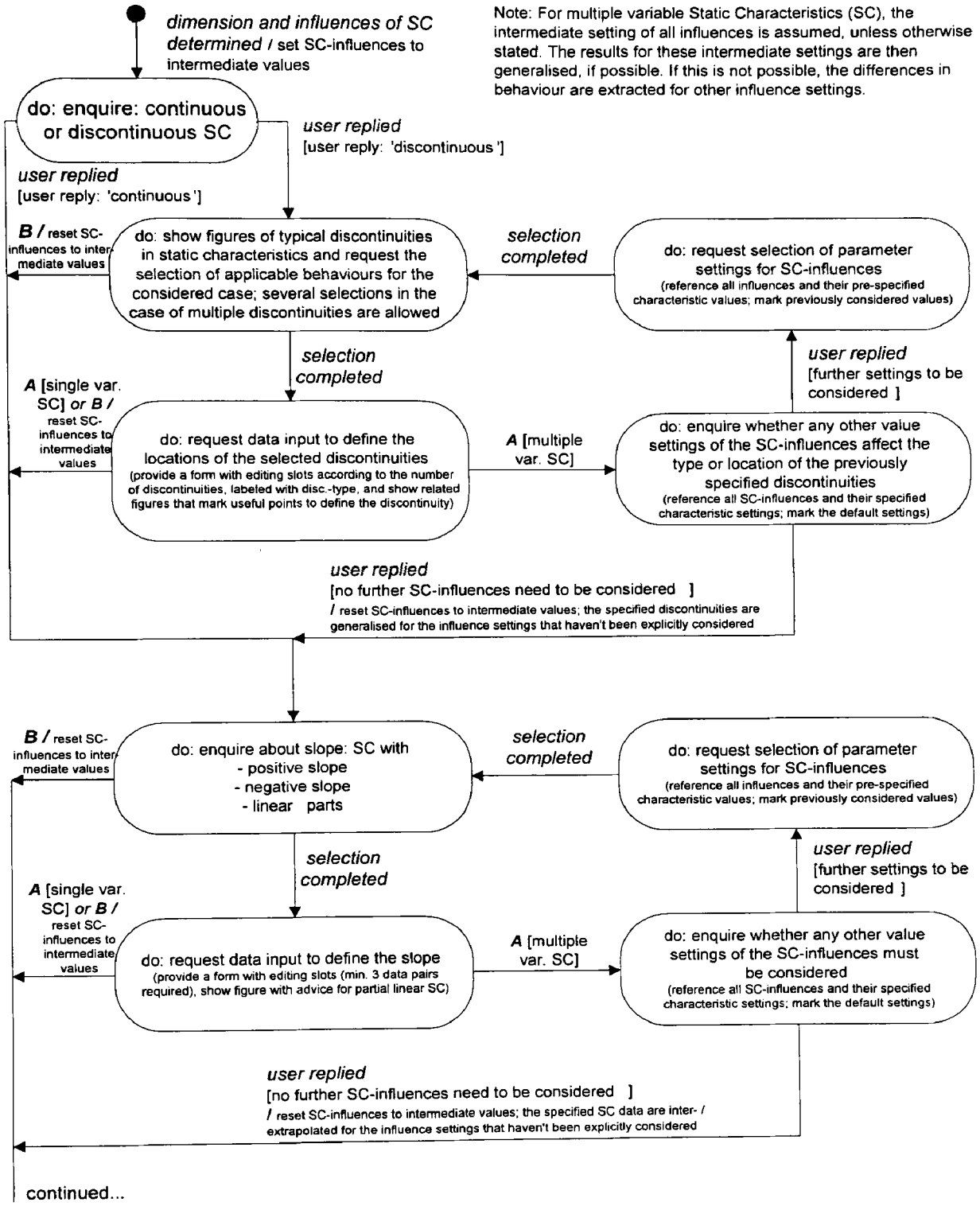


Figure 5-15: Sub-Sequence of NLs: Generalisation And Specialisation Approach

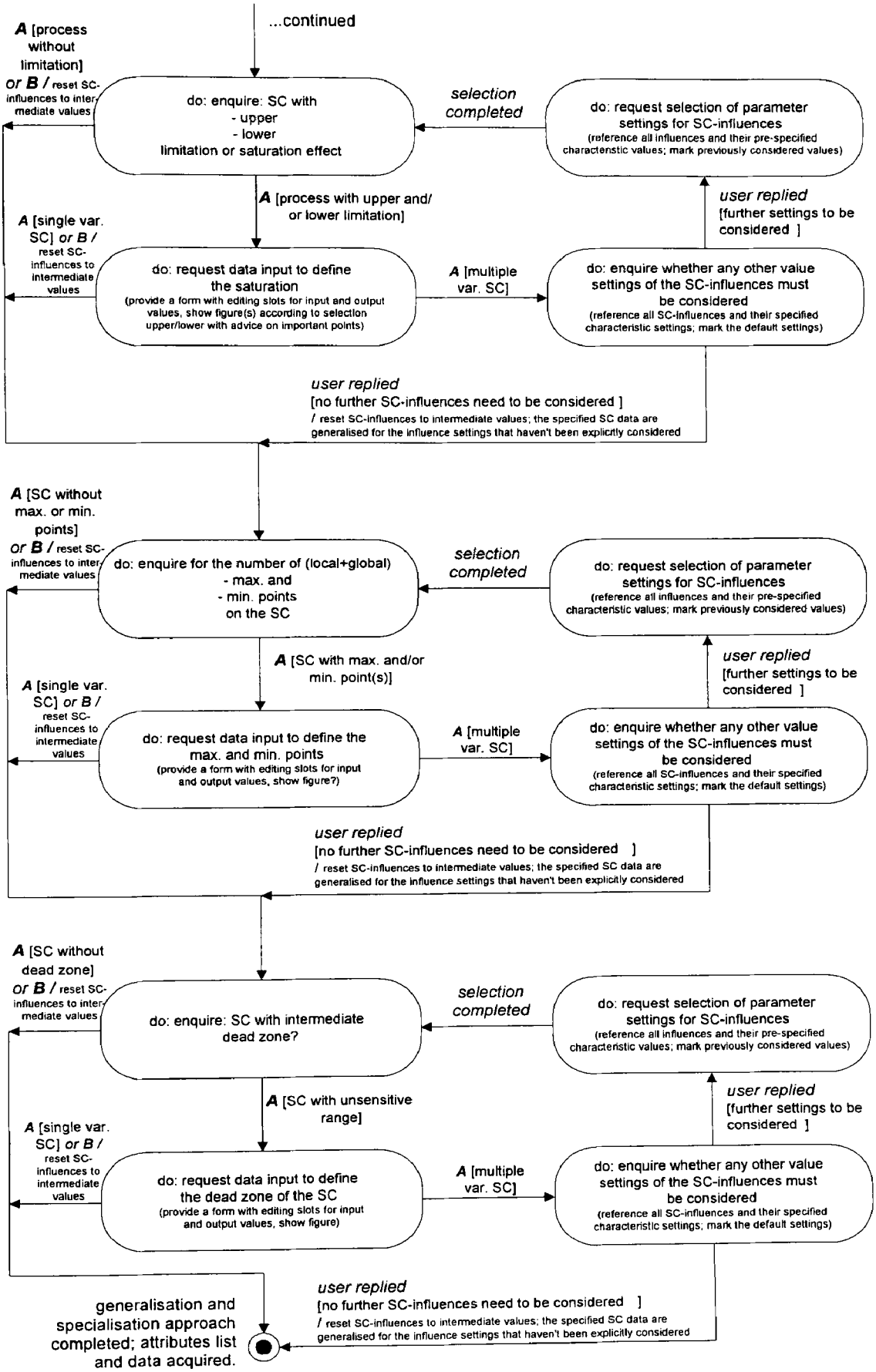


Figure 5-15: ..continued

Therefore, this sequence marks the bottom end of the heuristics based modelling approach and, hence, the end of the detailed design of the state diagram for the *MODEL*^{ing} approach.

5.5 Analysis And Design Of The Functional Model (Data Flow Diagram)

The *MODEL*^{ing} approach is mainly about the acquisition of knowledge and the translation of the user's experience into technical (i.e., systems science or control engineering) terms. Of particular importance is here the systematic user guidance and the structuring of the acquired knowledge, because the main difficulty of the practical user is to judge the importance of pieces of information, to structure them, make them usable and finally to use them for control engineering purposes.

Rather than actually translating all process knowledge in the sense of processing the input data to yield a usable format, the *MODEL*^{ing} approach aims at guiding the user in a way that enables him/her to specify the information in a directly usable form. The sequential modelling methodology (see OMT State Diagram) that puts the user into the appropriate context, together with the graphical user interface (GUI) act therefore as 'translators'. Using simplified, control oriented terms, the user is encouraged to reflect his/her own experience from a control engineering point of view and to respond accordingly. Since the majority of the knowledge acquisition approach is directly accomplished in this way, which does not require processes to transform data values, the Functional OMT Model of *MODEL*^{ing} is very simple.

Essentially, each requested piece of information is associated with a particular object or slot. Whenever a piece of information is provided by the user, the appropriate instance is created and its slots are filled with the data (Figure 5-16). By assigning the information to a particular class and slot within the object structure, the first essential step of ordering and structuring the knowledge is accomplished. After the completion of the actual acquisition process, the model information of the component is distributed over several instances of distinct process information within the object structure. On the basis of this structured knowledge representation, a summary is generated, which contains in report form (ASCII) the assorted component information. Similarly, simulation code is generated which includes the translation of locally valid linear SISO process models into nonlinear multivariable models according to the "fuzTF" approach. Both the summary of structured model information and the simulation code are passed on for further use to the other CACSD modules currently under development. Additionally, the model summary can directly be analysed by the control expert and fed back for information purposes to the practical user.

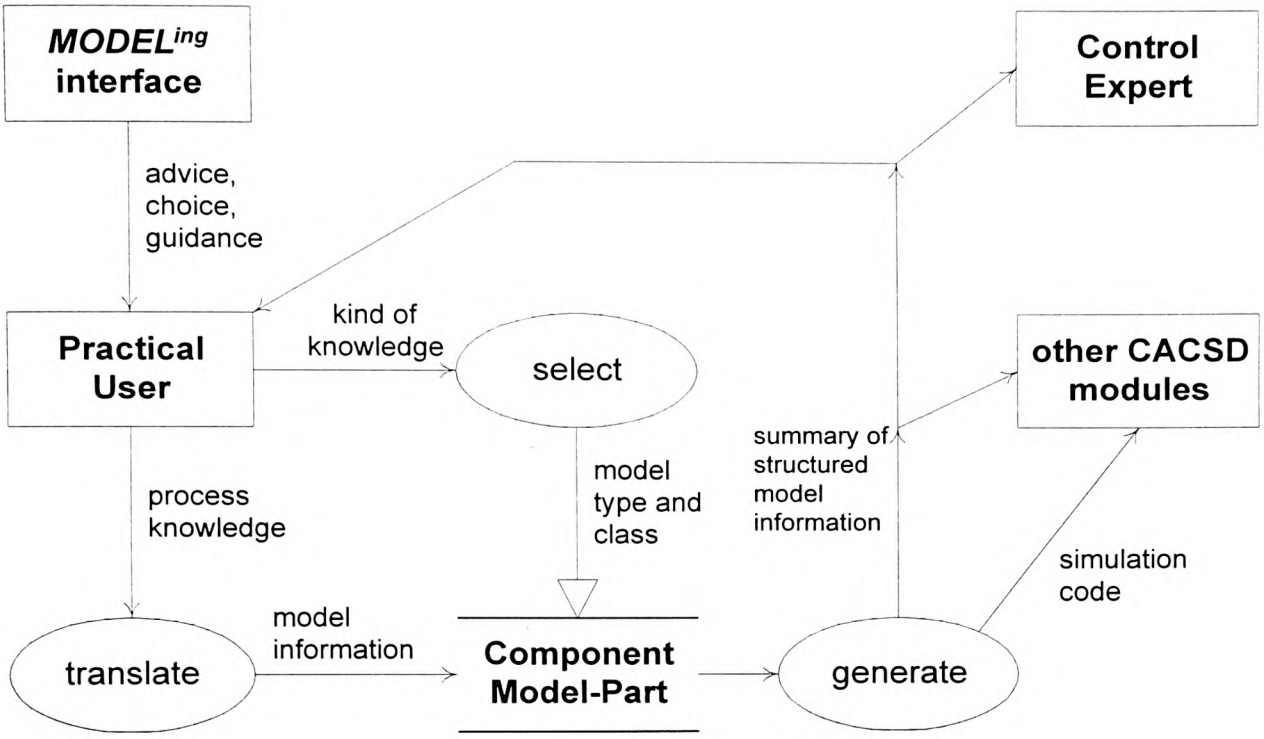


Figure 5-16: The *MODEL*^{ing} Data Flow Diagram

This coarse, 'overall' data flow diagram (Figure 5-16) illustrates the important concept of knowledge extraction that is described above. Further detailed levels of the functional model are, however, not required for this project as they would only feature a repetition of the concept, showing instances of all individual classes that are already detailed in the object model. Examples for some of the few exceptions, where input data requires explicit processing *before* being collected in different instances and their slots are:

- the deduction of attributes from the linear SISO modelling sequence
- the calculation of transfer function parameters from characteristic step response parameters in the linear SISO modelling approach
- the automatic generalisation of results in the nonlinear statics modelling sequence (part 5.4.10).

While data flows and transformations play therefore a less significant role in the *MODEL*^{ing} approach, the provided data and information influence significantly the flow of control in the knowledge acquisition sequence, as shown in the dynamic model in section 5.4.

5.6 Chapter Summary

The frame-based knowledge representation was selected as an efficient means to structure the acquired knowledge. A flexible frame concept which is based on an association of multiple instances per modelled component has then been defined to enable the frame-based representation to adapt to the amount of provided information.

For the analysis, design and documentation of the integrated KE approach, the Object Modeling Technique (OMT) by Rumbaugh et al. [34] was selected. Rumbaugh et al. point out, that not all OMT model types need to be developed in full depth for all applications but that, in fact, emphases should be put according to the given problem and that the tools should only be applied as far as it is helpful for the specific task. Further it is noted that "one way to characterise an application is by the relative importance of its object, dynamic and functional models".

Among the three OMT-models, the dynamic model is obviously of salient importance for the design of the *MODEL*^{ing} methodology, which is also expressed in its relative complexity and detail level. In this work, advantage was therefore taken of the flexibility of the OMT approach for system design to move the state diagram centre-stage.

Again, it should be stressed at this point that 'design' referred in this chapter to the design of the knowledge engineering **methodology**, the approach as such rather than the design of a program, which serves mainly the purpose of proving the applicability and validity of the approach. Apart from the methodology, another important part of the contribution is the design of the self-explanatory graphical user interface, which is, although described in the following chapter on the implementation, considered as somewhat distinct from programmatical issues. In fact, the knowledge engineering methodology stands, together with some key illustrations from the GUI, for itself in that it could, for example, just as well be summarised in an illustrated, easy-to-use booklet that would guide the user through the sequence and refer to the underlying equations and look-up tables. Nevertheless, apart from several disadvantages of such a 'paper version' of this approach, anything other than a software implementation would be simply out of date.

Although it might be possible to alter some parts of the KE sequence slightly without major effects on the overall approach, it should be pointed out, that great care has been put into the design and optimisation of the sequential flow of control within this knowledge acquisition sequence in order to address the particular needs of the envisaged user group:

- With the aim to be self-explanatory, the succession of modelling steps was designed to bring the user more and more into the subject without too lengthy explanations.
- Avoiding parts of the approach that are, according to the previously provided information, redundant facilitates a relatively short and efficient modelling sequence.
- And finally, the user stays despite the guidance in control of the approach.

Special care has also been taken to incorporate the handling of complex dynamic systems according to the "fuzTF" systems view. This design consideration ensures a smooth integration of the two main contributions of this work: the complete information required to specify a fuzTF model of the component can be acquired very comfortably so that all matrices and blockdiagrams can be automatically generated. To put these theoretically possible links into practice and, hence, to validate the contributions, a *MODEL*^{ing} prototype was implemented as will be described in the following chapter.

6. Prototype-Implementation Of The Knowledge Engineering Approach

The analysis and conceptual design phase of a project, which is documented in the three OMT diagrams [34], is according to Rumbaugh's OMT methodology, succeeded by the implementation design phase in which the diagrams are revised from a pragmatic point of view and decisions with respect to hard- and software organisation, data and resources management are made.

This work, however, focuses on the new knowledge engineering *methodology*, as it has been expressed before. The purpose of its prototype implementation is mainly the validation and verification of the methodology as well as the elaboration of some aspects that are essential for the realisation. Most implementation considerations which would be essential for a final implementation are therefore of secondary importance for this prototype and only those aspects that are directly linked to the application and applicability of the methodology are discussed in more detail in this chapter.

6.1 The Programming Environment

At a very early stage of this work, the research group decided that all modules within the overall CACSD approach in the collaborative project should be implemented on a personal computer (PC). The main reasons for this decision were the widespread availability of this hardware platform in industry and its low cost.

In the selection of the programming environment, its suitability to the rapid prototyping of graphical user interfaces was a major consideration. Another particularly important requirement was object orientation in order to facilitate the direct translation of the conceptual design into the program and also to be able to take advantage of the various benefits this paradigm offers [34]. Due to the advantages of *event-driven* systems with respect to flexibility of control patterns, simplified and powerful mapping of events to program constructs as well as improved modularity and error handling, it was decided that this type of software control should be adopted as far as possible instead of a purely *procedure-driven* concept in the implementation of the *MODEL*^{ing} methodology.

The general expert system tool Kappa-PC [124] was chosen as the programming environment. It allows not only for event- and procedure-driven implementations but also rule-based software

control. This flexibility proved to be very convenient for the realisation of the different modules within the knowledge engineering approach. A major reason for the selection of Kappa-PC was, however, that it is probably one of the most consistently object oriented programming environments for PC. Furthermore, Kappa-PC is ideally suited to prototype development, firstly, because it features simplified user interface programming facilities and secondly, because it is an interpreter based development environment. Each new piece of program code can therefore be instantly tested which is extraordinarily helpful.

The programming language used in Kappa-PC is called "KAL". KAL is a high level language which can be compiled into ANSI-C code using the KAL compiler. The standard development environment is specifically aimed at rapid prototyping and is supplied without the compiler. Kappa-PC offers different possibilities of integration with programs developed in other languages such as Visual Basic, C or FORTRAN as well as with commercial products such as spreadsheets or database programs.

Kappa-PC is only the programming environment; all features of the implementation described in the following sections - including all data handling - have therefore been purpose-built within this work.

6.2 Implementation Of The Object Model

Taking advantage of Kappa-PC's object orientation, the *MODEL*^{ing} object structure discussed in Chapter 5 was implemented with only minor changes in the program. Figure 6-1 shows the object browser directly from the prototype in Kappa-PC.

The main difference between the original object diagram (Figure 5-4) and its implementation (Figure 6-1) is the focus of the latter on 'inheritance' relationships. This difference is, however, mainly the result of a superficial reorganisation: Although some formerly 'aggregation' relationships have been replaced for efficiency reasons by 'inheritance' relations (where applicable), the two object model versions do not contradict each other. "NLSC" (nonlinear static characteristic), for example, is both *"a kind of"* (static) MODEL (i.e., *inheritance*; in the implementation) and *"a part of"* (the overall) MODEL (i.e., *aggregation*; in the original object model). Furthermore, all essential aggregations and associations (example: MODEL *has* Terminals) which are part of the original object diagram but not shown in the simplified implementation version are programmed as 'Object Slots' in Kappa-PC. Rather than handling text or numerical attributes, like 'ordinary' slots do, the slots of the type 'Object' associate the object which owns the slot with the object whose name is entered in that slot.

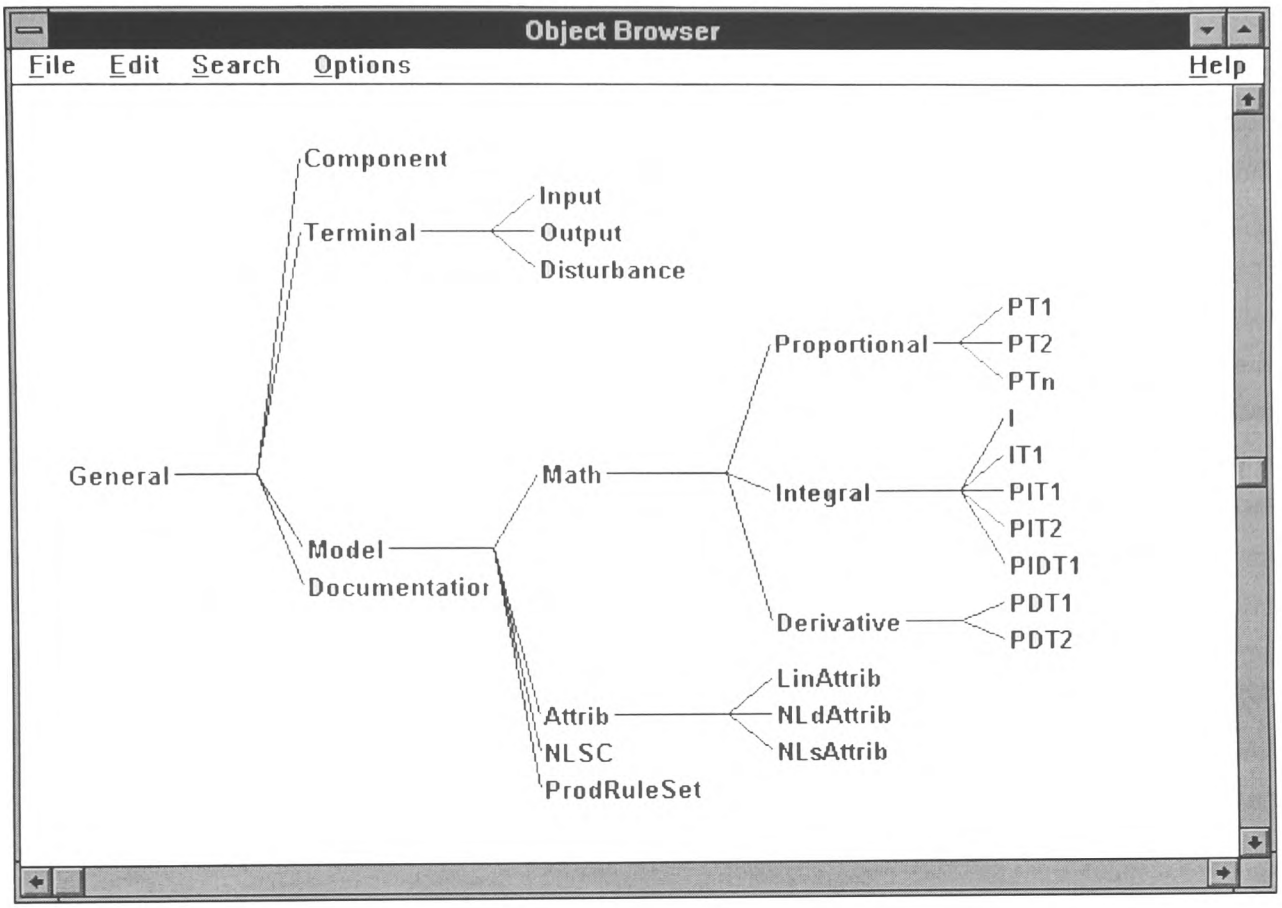


Figure 6-1: The Object Model Of The *MODEL*^{ing} Prototype

The design decisions that led to the final, implemented object structure considered, in particular, the realisation of the flexible and efficient knowledge representation that was introduced in the previous chapter. This flexible frame-based knowledge structure (cf. definition in Chapter 5) benefits greatly from the object oriented approach. By creating only the required number of instances of classes like input and output terminals or different linear and nonlinear model components and combining these instances via associations, the actual size of the overall frame always adapts to the particular needs. Some more specific aspects of the flexible frame and its handling are described in the next section.

6.3 Data Structure And Handling

As indicated above, this section describes the "application" of the object structure to one of its main purposes, the systematic and flexible handling of data.

The different pieces of acquired information on a particular component are handled as a collection of instances in the object tree. In the course of modelling, new instances are automatically created (i.e., "instantiated") as needed by the program, which assigns also the instance names and ensures their uniqueness.

A "coding" system for instance names was devised to handle the complexity resulting from the possible accumulation of instances per process model. This systematic naming of instances is a very important feature for trouble shooting during system development and also for the expert user of the program since it facilitates the recognition of each instance, the kind of information it contains and what it relates to. The originally developed coding included component name, version number of the component model, the type of knowledge it was derived from, the output it relates to, and a number related to the operating point it refers to. This code, however, was considered too complex for its purpose and therefore abandoned for a far simpler one which contains only the name of the output it refers to, the type of knowledge it was derived from and a simple, unique counter number. "PressureH6", for example, would be the instance representing the sixth local model that was created from heuristic knowledge ('H'), describing the behaviour of the output 'Pressure'. In future extensions of the implementation, 'C' will denote instances derived from causal knowledge and 'P' those derived from probabilistic knowledge.

The user does not have to remember the names of instances in order to edit them since the program accesses automatically the correct instance according to the user's selection of the operating conditions. Furthermore, the program ensures that there is no more than one instance containing a particular type of process information for a specific operating condition.

Each output instance plays an important role in that it has multiple value slots (i.e., lists) that keep track of all instances with behavioural information on it. Such instances with behavioural information could be, for example, linear single variable transfer functions that are locally valid for a particular operating condition or a set of attributes that describe the static and dynamic properties. Special sorting functions are responsible for arranging the entries of instance names in the multiple value slots ('track-keeping slots') of the output according to the validity of the related local model instances at different operating points. Each multiple value track-keeping slot is equivalent to a column of an imaginary matrix (Figure 6-2). The dimension of this matrix depends on the number of influences (apart from the MAIN input). In its current prototype-implementation, *MODEL*^{ing} handles two such influences with up to 9 operating points each. A global model is therefore currently made up of up to 81 locally valid models whose instance names are sorted into 9 track-keeping slots with 9 entries each, making up a 9 × 9 imaginary matrix. A separate multiple value slot keeps track of

the correct sequence of the columns in the matrix. Individual 'imaginary matrices' exist for the different types of process information (dynamic math models, purely static information, attributes, etc.). Rather than using lists making up imaginary matrices, 'real' matrices should normally be used for the systematic ordering of instance names, if the implementation environment allows for multi-dimensional slots in the object structure. The complexity of the instance sorting functions, however, would not be affected.

$$\begin{bmatrix} inst.C & inst.E & \dots \\ inst.A & inst.D & \dots \\ inst.F & inst.B & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

Figure 6-2: The Handling Of Multiple Model Instances

Each instance name is allocated to a matrix position according to the validity of the instance with respect to particular settings of the influences: each row relates to a specific setting of the first influence parameter and each column to a specific setting of the second influence.

The central 'track-keeping slots', together with the sorting functions are therefore essential in the process of converting the largely unstructured information provided by the user into both a structured report on the component's characteristics and simulation code, including correctly compiled parameter matrices. Thus, the information sorting in the *MODEL*^{ing} program can be considered as a two-step process. In the first step, the information is allocated appropriately into the object structure (object-oriented sorting), whereas the second step (i.e., report and code generation) is responsible for combining the information that is distributed over the object tree as it causally 'belongs' together (causal sorting). The causal sorting involves, for example, the combination of information referring to a particular output and the arrangement in the sequence of specific operating conditions (example: aggregation of parameter matrices with each matrix position originating from different instances).

Together with the output instances, all 'track-keeping slots' are automatically created only as needed - they inherit their names and properties from the output *class* in the instantiation. To ensure that the overall transfer function structure covers all occurring operating conditions also in case of order changes, two further slots per output keep track of the highest occurring numerator and denominator order.

A separate 'Documentation' instance handles all general information on the component, like the version number, a statement of the model purpose, verification and validation tests and test results, comments and the like (cf. Barker et al. [24]).

After completing the modelling, all instances with their slots and slot values that define the component are saved to a file with the component's name. At the same time, the name of the component is transferred to different slots of a central "Library" instance which is crucial to the browser-supported information retrieval. These library slots are lists that are named according to the different process domains and process types which are specified for each component in the course of modelling. If the specified process type and domain of a modelled component are existing slots in the library, the component name is simply added to these lists. Otherwise, new lists (i.e., slots) are created first. The organisation of component names in the library is therefore comparable to the file system on a PC with main directories (equivalent to 'process domains') and sub-directories (equivalent to 'process types'), although there is an important difference: on a PC, the actual files are only referenced by the sub-directory, whereas the main directories contain only a direct reference to the sub-directories and the root directory refers to main directories. The library, on the other hand, can list all previously modelled components as well as the process domains. Within each domain, it can show the sub-set of components that belong to the particular domain and it can list the process types within that domain. Finally, an even smaller sub-set of all components is listed when a particular process type is considered. This system ensures the flexibility of the browser in the process of reducing the choice of components in a step-wise manner. Since the user is entirely free to categorise the components in process domains and types according to his/her individual needs, the advantages of this library system in comparison with the ordinary directory approach pay off especially when a different user wants to retrieve a component: this other user could have a slightly different perception of the considered domain - rather than having to search lots of sub-directories, he/she can search on intermediate levels.

Before modelling a new component, all instances related to the previously modelled component are deleted.

Deleting information that is referenced in another part of the component model or deleting process types or process domains for which example processes exist could lead to major problems in the program or unwanted loss of information. Various functions have therefore been programmed to ensure that only 'independent' information can be deleted. In order to delete a process domain, for example, it is necessary to delete first all process types that are allocated as categories to this process domain. Before a process type can be deleted, however, all process components of that type have to be deleted individually. Likewise, it is not possible to delete a locally valid instance within a component if this instance is already referenced in one of the 'track-keeping slots' as a part of the global component model. It is, however, possible to edit this locally valid instance, if required.

When exiting the program, the component library instance, which has been extended by the components modelled in the preceding session, is saved to file, while the *MODEL*^{ing} program as such remains unchanged. The updated library is automatically re-loaded at the next *MODEL*^{ing} program start.

Overall, the data handling procedures that run in the 'background', hidden from the user, are relatively complex due to the possible degrees of freedom of processes that can be modelled. Automating and hiding this complexity handling was not only a major aim of this research project but it is very sensible since the routine functions required to sort and structure information are very suitable for programming. This not only opens the door to modelling complex systems for inexperienced modellers but also frees resources for expert modellers to focus on aspects that cannot be automated. Furthermore, there is no merit in making the user acquainted with the internal structure and handling of data as long as the functions that translate between the user's 'picture' of the problem and the system's internal view are consistent and well tested.

In relation to the final model as it is exported for simulation purposes, the program-internal, object-oriented representation of the component model that has been described in this section is an intermediate model ('meta model'), which is neutral in the sense that it is independent of any particular simulation program.

6.4 The Graphical User Interface

The sequential flow of control, which is represented by the OMT state diagram (cf. Chapter 5), is central to the *MODEL*^{ing} methodology. Very closely related to this sequence of the knowledge acquisition procedure is the graphical user interface. It is the key to accessing the user's knowledge in the different stages of the procedure.

Unfortunately, the modelling sequence could not be fully implemented by the end of this research project. The focus was therefore put on the essential stages of the dynamic nonlinear multivariable modelling which includes, in particular, all information required for the generation of 'fuzTF' models that were introduced in Chapter 4. In the remainder of this section, the implemented GUI sequence is illustrated.

The 'Main Menu' (Figure 6-3) was designed as the "fix point" of the *MODEL*^{ing} approach. It is not only the initial and the final state of the sequence, but it is also accessed whenever the user wants to discontinue following a particular sequence (cf. Chapter 5.).

A hand-drawn picture in the centre of the screen indicates to the user what the program is about. Similar sketches appear at different stages of the modelling procedure; their purpose is not only to illustrate the concept of the approach (breaking down a multivariable global model into several singlevariable local models), but also to indicate that this approach is modelled on basic pencil-and-paper approaches. The latter aspect could be psychologically important for users who are scared at the idea of contact with a CACSD program.

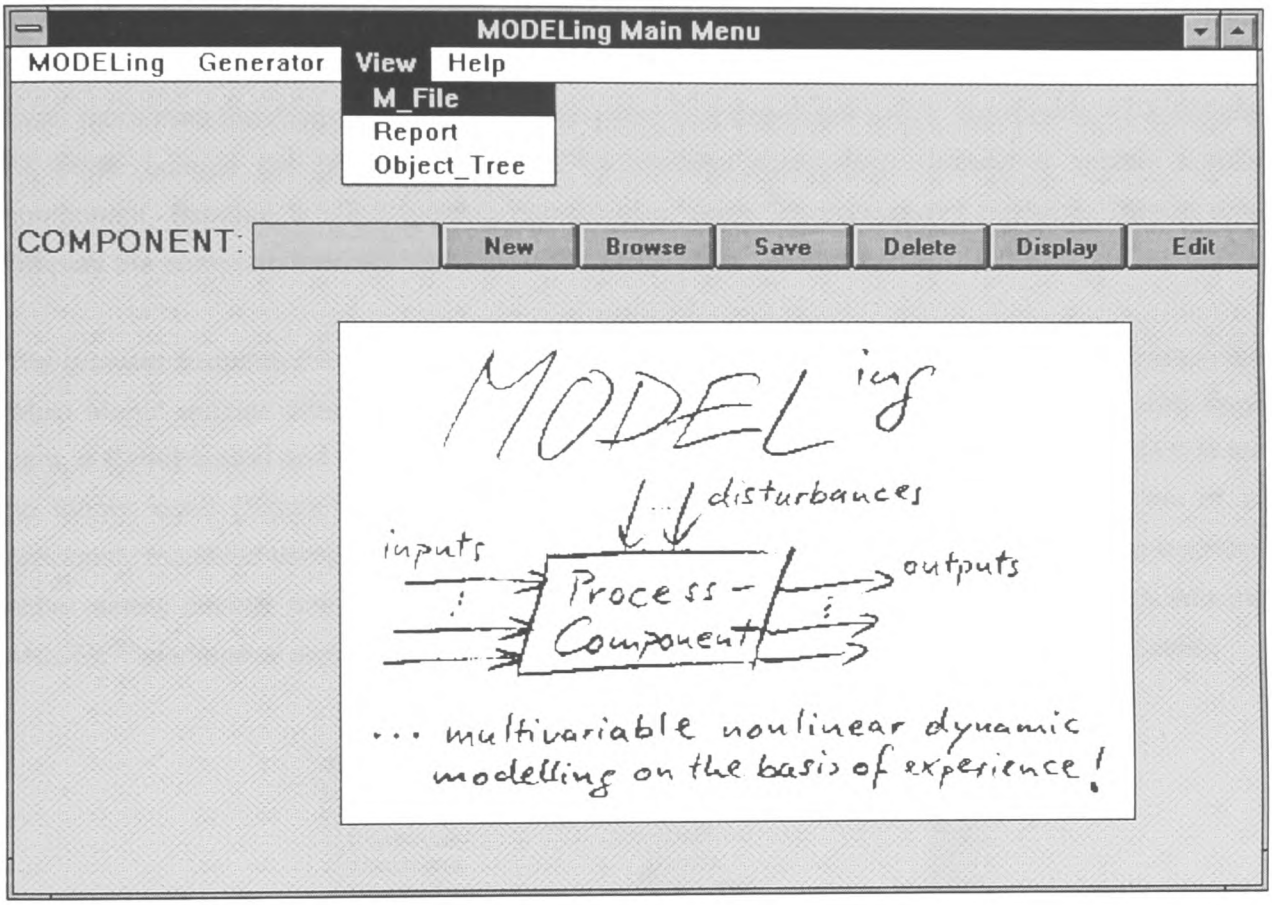


Figure 6-3: Main Menu

In the prototype implementation, the 'Main Menu' (Figure 6-3) features the following options:
Firstly, a menu bar with the items ***MODEL^{ing}***, **Generator**, **View** and **Help**, which lead the way to different functions, is provided.

MODEL^{ing}	**Generator**	**View**	**Help**
About...	MATLAB Code	M-file	Info
Set Path	ASCII Report	Report	
Exit		Object Tree	

"Set Path" is used to define the program- and working-directory. The **Generator** functions create files for the export of modelling results. Via the **View** functions, modelling results in the form of M-files or ASCII Report files can be checked. Furthermore, the program-internal model representation in form of the Object Tree can be viewed. The **Generator** and **View** functions are discussed later in this chapter. Under **Help** "Info", the user is advised to request specific help in the different modelling stages by pointing either anywhere in the window or at an item in question and pressing the right mouse button.

Even more than the menu bar, it is the "button bar" that draws the user's attention to its functions. All these buttons are directly related to the process component. Starting to model a **New** component, **Browse** through existing components, **Save** the component model or **Delete** one, **Display** the component details and **Edit** an existing model are the options.

The browser supports the systematic retrieval of previously modelled components. It appears in the 'Main Menu' window after selecting the **Browse** button and consists of two list boxes with scroll bars, a **Reset** button and two maintenance buttons for deleting process domains and types that are no longer used (Figure 6-4). Whereas the right list box gives an alphabetic overview of all previously modelled components, the left one lists all process *domains* as categories. The user can either select directly one of the components from the right list to load the model data into the *MODEL*^{ing} workspace or firstly narrow the choice of components by selecting a process domain.

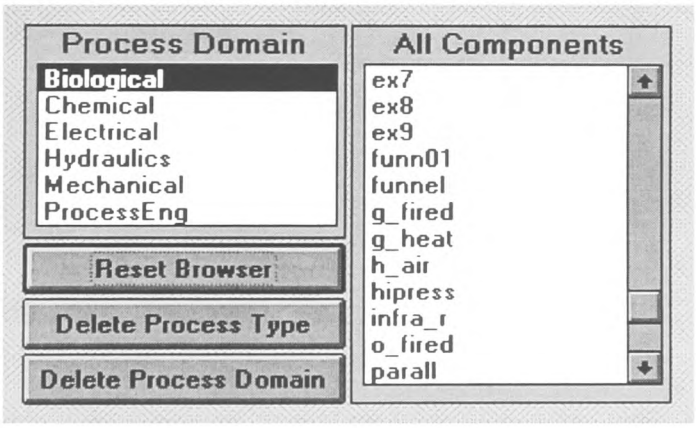


Figure 6-4: Browser Support

After choosing one of the process domains, the component list on the right shows only the sub-set of components belonging to that particular domain, which is also indicated by the heading of that list box ('<Domain> Components' rather than 'All Components'). Instead of showing the process

domains in the left list box, the process *types* within the selected domain are now shown on the left (Figure 6-5). Again, the user can either select directly one of the components or reduce the remaining choice further by selecting one of the process types (Figure 6-6), before the final decision for the appropriate component is made. **Reset Browser** allows a re-start of the selection process by resetting the browser to its starting condition. This efficient, stepwise selection support facility becomes increasingly valuable the more components have been modelled before.

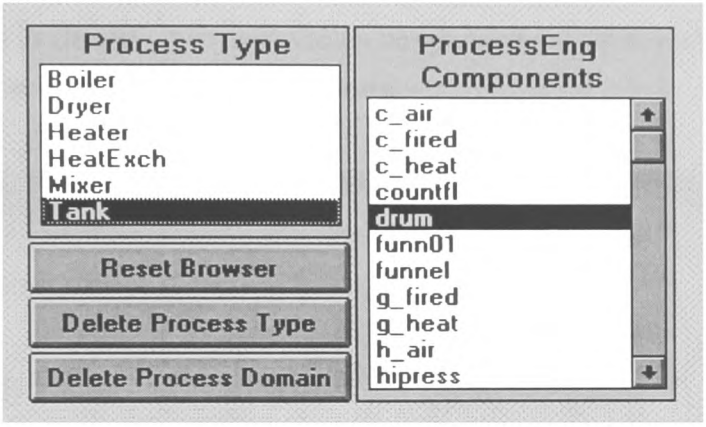


Figure 6-5: Browser After Domain Selection

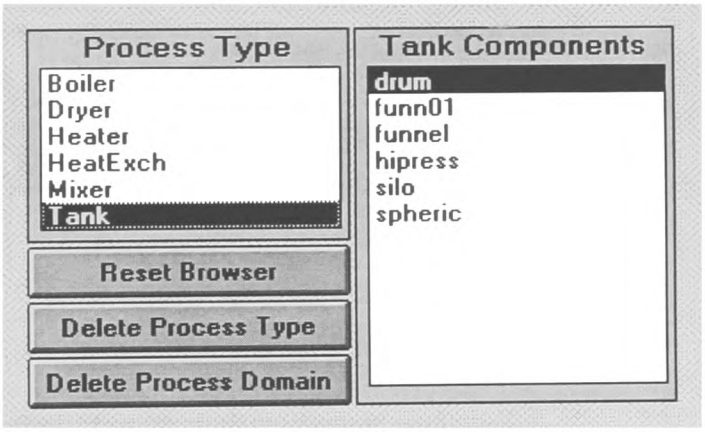


Figure 6-6: Browser After Type Selection

Whenever a component is selected in the component list box on the right, the appropriate data is loaded and the browser disappears. Interface files (MATLAB-file or ASCII report) can now be generated and viewed or the component model is edited after pressing the **Edit** button. With the selection of **Edit** or **New**, the actual modelling sequence commences with the 'Isolation of the Component' (cf. Chapter 5.). The sequence for editing a component is very similar to the one for modelling a new component, which is detailed now.

After selecting the option **New**, the user is requested to specify a component name. The program allows only names that have not been used before. The version handler, which appends automatically the number '1' to a new component and increments it whenever the model is edited in order to maintain intermediate modelling results, has not yet been implemented.

In the following illustrations, the funnel tank that was previously used to demonstrate the 'fuzTF' approach introduced in Chapter 4 is here shown as an example. At the component name prompt, the user therefore specified "funnel" in this example.

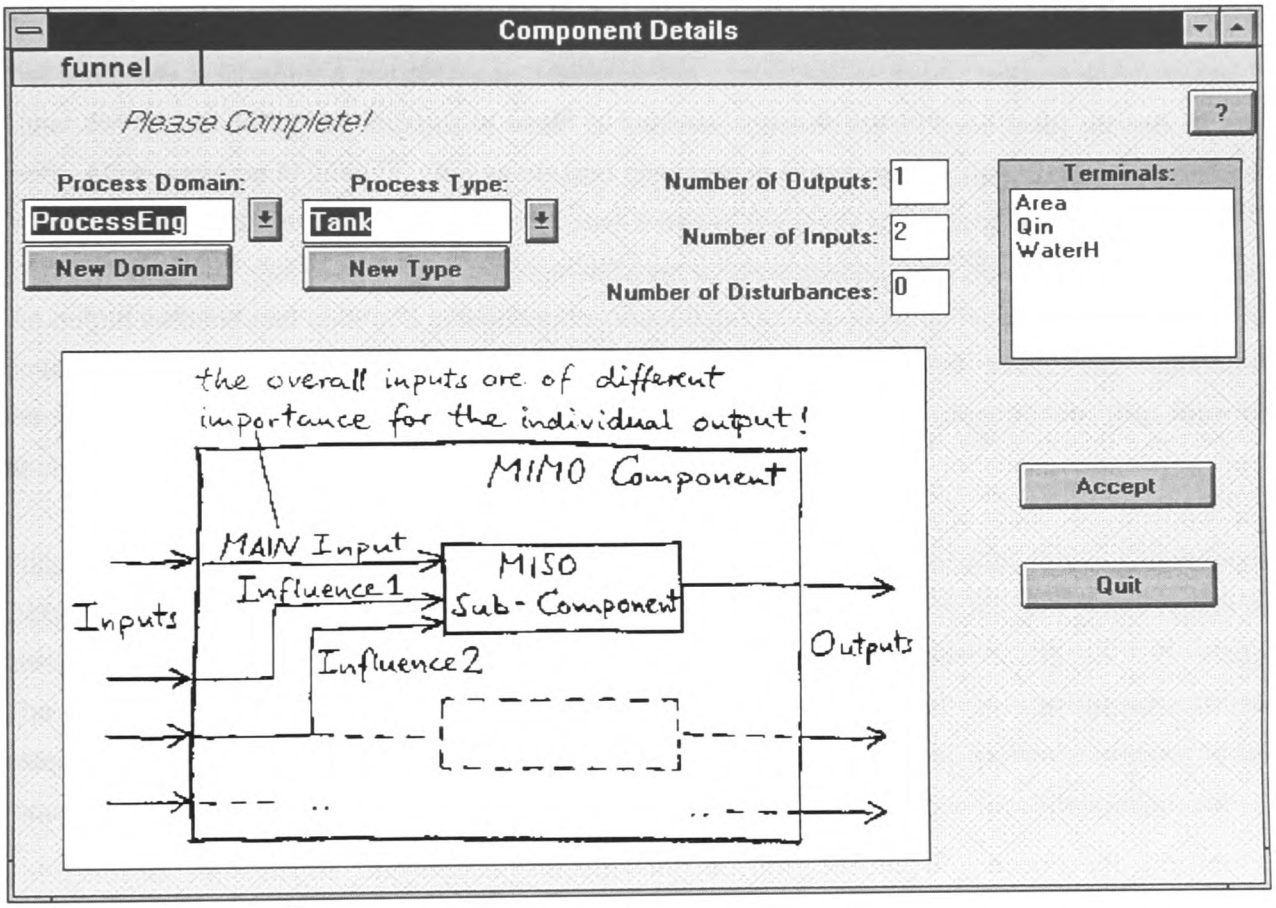


Figure 6-7: Component Details

In the 'Component Details' screen, which is shown directly after specifying the component name, the user is requested to define the process *domain* and process *type* for future browser-supported retrieval of the component (Figure 6-7). The user is entirely free to select among previously defined process domains and types or to define new categories according to the individual needs. All newly defined process domains and types are automatically supported in the browser after saving the component.

Three further input facilities are provided in the 'Component Details' screen: the number of process inputs, outputs and disturbances must be defined. According to the specified numbers, the user is prompted to give names to all terminals - the default names are 'Out1', 'Out2', 'In1', etc. It is, however, advisable to replace these defaults by meaningful names to simplify all stages of the modelling as well as the later use of the generated model in a simulation environment (in the funnel example: output "WaterH" and inputs "Qin", "Area"; cf. Figure 4-12). After completion, the **Accept** button is pressed and the user is in an interactive form-filling procedure requested to provide more detailed information on each terminal. This information includes essential parts like the symbol of the variable, the units, a list of typical operating points and max. / min. process limits as well as additional information (in the funnel example, the operating points for each of the three terminals that are given in Chapter 4 are entered). The essential information is clearly marked and necessary either for the modelling as such or in order to maintain consistency in the overall context of this work. After applying *MODEL*^{ing} , the generated component model can be transferred to the block editor of a simulation environment where it forms a part of the structured overall plant specification[1]. In this block editor, the connections between different process components must be consistent, i.e., the output variable and units of a process component must be the same as the input variable and its units of the following process component to which it is connected. The additionally requested information is very useful - particularly for the design of identification experiments (e.g. type of applicable test signals) - but does not affect the modelling as such.

Following the completion of terminal information, the **Continue** button appears in the 'Component Details' window. It is the starting point of the nonlinear dynamics modelling sequence which is labelled 'NLd' in the conceptual design of the *MODEL*^{ing} approach (Chapter 5, Figures 5-7...5-10). The first two states, NLd1 and NLd2 (top of Fig. 5-7, and Fig 5-8), which are concerned with the selection of the output variable as well as the associated MAIN input and further influences to be focused on in the following part of the repetitive procedure, are handled in a form-filling sequence.

In the funnel example, the only specified output, "WaterH", is selected here and the input "Qin" is defined as the MAIN input. The output "WaterH" is also selected as the first nonlinear influence on the dynamic relation between "Qin" and "WaterH", while "Area" is specified as influence 2 (compare with Figure 4-13).

[1] the detail level of the component to be modelled using *MODEL*^{ing}, however, depends solely on the user: the available knowledge, the effort he/she wants to put into the modelling and the requirements of the application for the process model - therefore, it is also possible to model the complete plant as a single component in *MODEL*^{ing}

After this short form-filling sequence, a new screen appears. This new screen, titled 'Influences: Combination of Local Operating Conditions' (Figure 6-8) marks the beginning of the 'Main Body' of the nonlinear dynamics modelling sequence (NLd3: middle of Fig. 5-7, Fig. 5-9, Fig. 5-10).

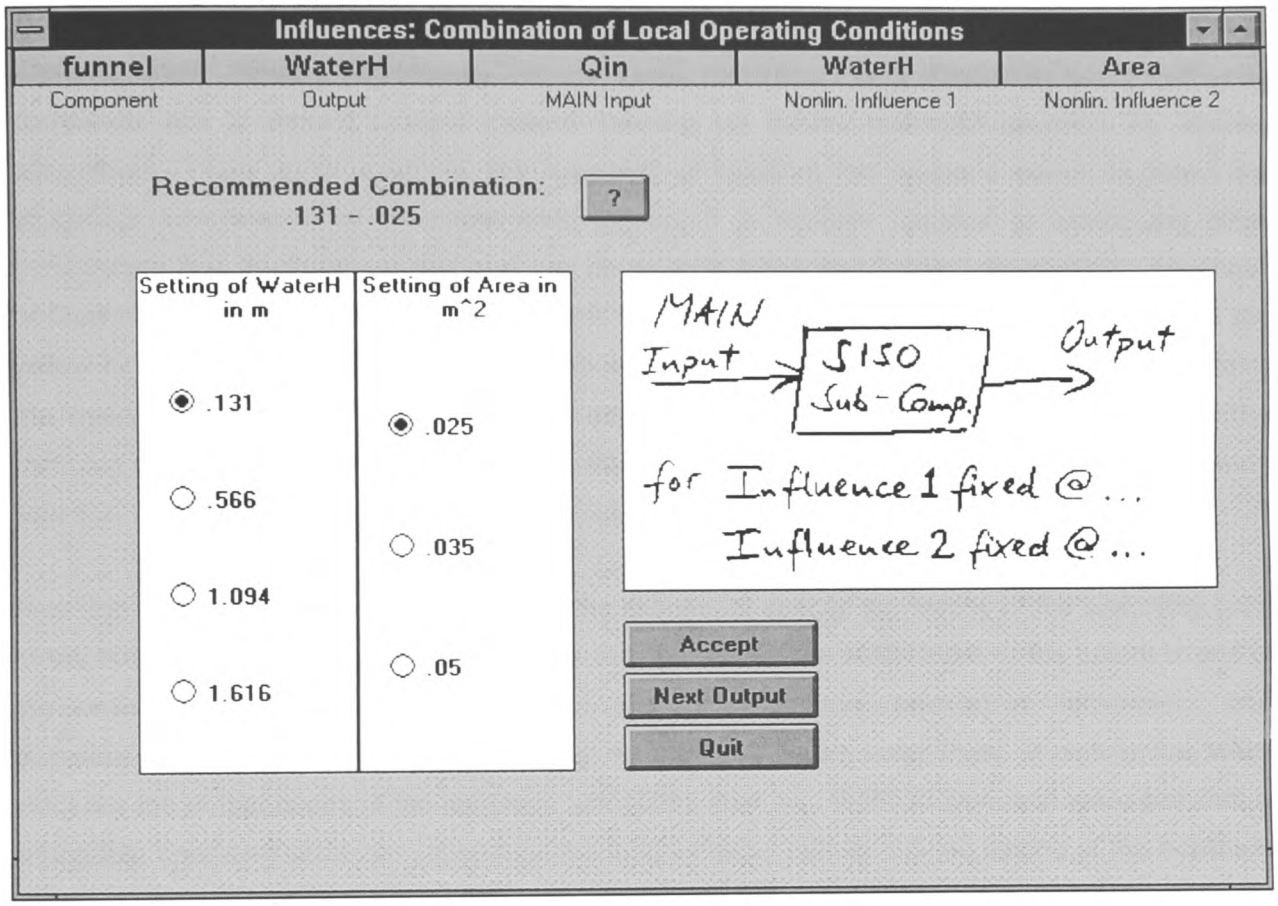


Figure 6-8: Influences: Combination Of Local Operating Conditions

'Influences: Combination of...' is the interface to control the repetitive local single variable modelling. By selecting a combination of typical settings of the influences, the nonlinear multivariable component is broken down into linear single variable models, which represent the behaviour of the component only locally for the fixed condition of the influences. The options in the selection boxes are the user-specified typical operating conditions of the influences which have been acquired in the preceding form filling approach. Figure 6-8 shows the typical settings of the nonlinear influences "WaterH" and "Area" that are familiar from Chapter 4. After completing the 'local modelling' sequence, which follows the screen in Figure 6-8, the flow of control returns always to this interface so that a new combination of influences settings can be selected and the local SISO modelling is repeated for the new condition. Above the selection boxes for the parameter settings, advice is given on which influences combination to select next. When all SISO modelling is completed (i.e. all influences combinations have been considered), the user is advised to select the **Next Output**

variable. The '**?**' button is equivalent to pressing the right mouse button anywhere else on the screen: the help message *"Select the next conditions of the influences to be considered in the modelling of the MAIN input - output relation and confirm with <**Accept**>!"* appears.

The 'Influences:...' - screen shows the typical characteristic of all screens in the prototype implementation: "limited functionality without being restrictive". It is *limited* to avoid confusing complexity and to ensure straight forward handling yet avoids *restrictive* guidance by offering alternatives. There is, for example, the possibility of following the system's advice to select the suggested combination of influences settings, but it is likewise possible to select any other combination first. Furthermore, the user can move on to focus on another output and its influences or **Quit** this sequence and return to the 'Main Menu'. Another typical and important feature is the yellow highlighted status bar at the top of the window which should ensure that the user maintains the overview with respect to his/her position within the multivariable modelling process. The meaning of each entry in the status bar is briefly annotated in this particular screen - in the following screens, this explanation is given only on request through the right-mouse-button-help function.

Pressing **Accept** in 'Influences:...' prompts the request to specify the setting of the operating point of the MAIN input and the associated output setting. If either the MAIN input or the output is one of the nonlinear influences, the setting of this variable has already been selected in 'Influences:...' and therefore only the remaining variable setting must be completed accordingly. If neither the MAIN input nor the output is one of the nonlinear influences, then any MAIN input-output value pair that is a possible operating point can be entered since this means that for a fixed setting of the nonlinear influences the relation between MAIN input and output is linear throughout the operating range. In the case of the funnel example, the operating point of the output ("WaterH") is already pre-set from the previous window, since it is at the same time the first nonlinear influence. Only the appropriate setting of the flow "Qin", the MAIN input, must therefore be completed here.

The local operating condition of the multivariable process component is now fully defined and information on its dynamic behaviour is acquired for these constraints (cf. sequence in Fig. 5-9, Fig. 5-10):

Firstly, the modeller is requested to give an estimate of the process dead time, if applicable (the acquisition of information on limited rates of change has not yet been implemented). By separating the information on the dead time, the following dynamic modelling can be made without considering it, which simplifies the approach.

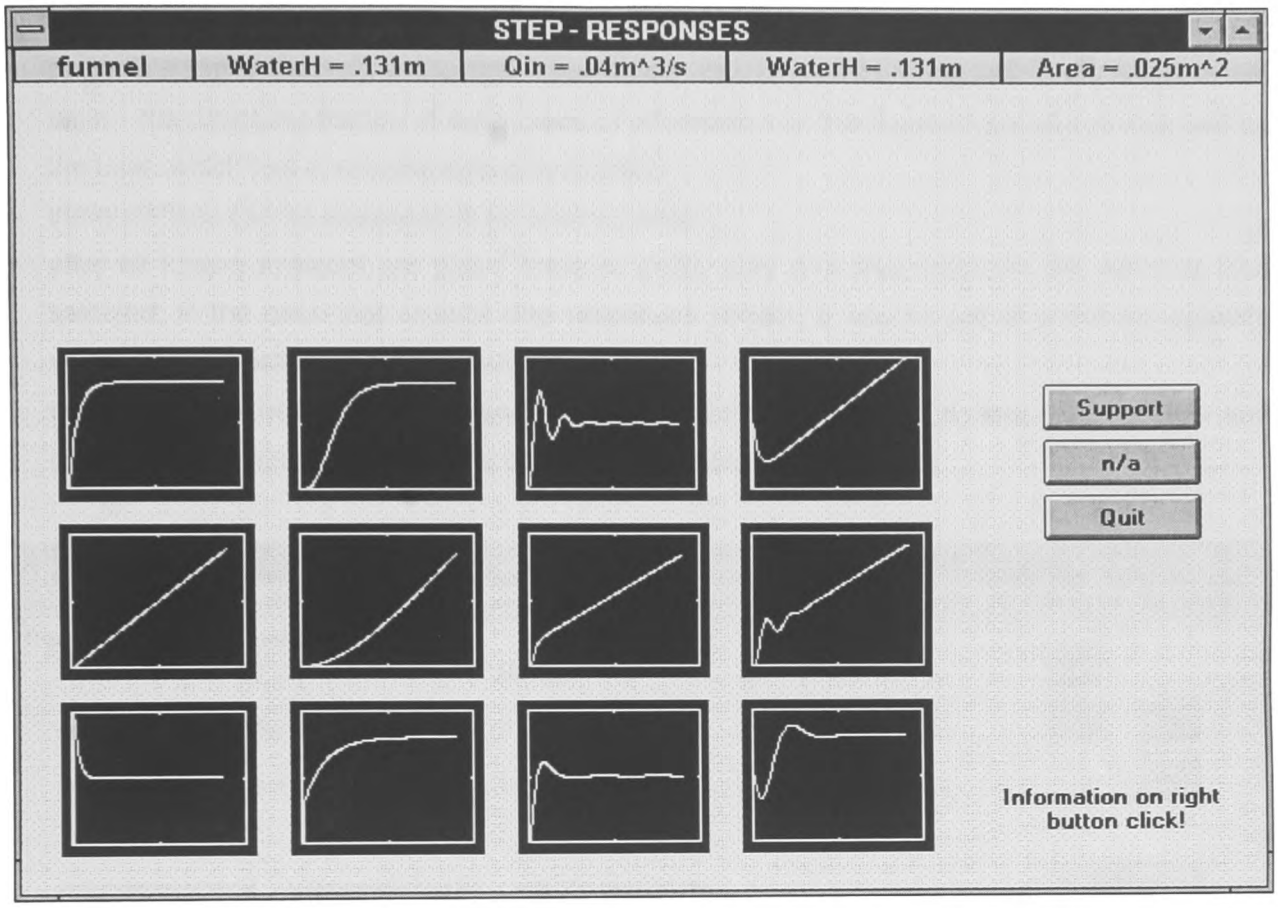


Figure 6-9: Step Response Types

In the following 'Step-Responses' window, Figure 6-9, the status bar is now completed with the settings of the influences as well as those of the MAIN input and output that have been specified for the considered operating point. This modelling step makes use of a graphical representation of step responses to access the process expert's understanding of the temporal behaviour of the process. The advantages of using step responses for this purpose have been discussed in Chapter 5. Without being bothered with theoretical details, the area engineer gives indirectly a very detailed piece of information by selecting the step response graph that matches the real process behaviour best. To address the situations, where a user has problems with making this decision, he / she is assisted by the decision support facility (Figure 6-10).

With the decision support facility, the choice of possible step responses is incrementally narrowed. Its main characteristics are as follows:

- firstly, the three most important questions for the decision are explicitly asked
- the answers are either 'yes' or 'no' - in the *MODEL*^{ing} program, a 'yes' or 'no'-button is pressed; the selected option remains on the screen, the other disappears

- in the case that an answer is unknown, the question can remain unanswered
- every answer is instantly evaluated: inapplicable step responses disappear from the selection table - the direct implication of each piece of information on the decision process is watched by the user, which has a valuable educational effect
- the questions can be answered in an arbitrary order
- after all known answers are given, there is ideally only one step response left which is then selected; in the case that several step responses remain, a second set of questions appears when 'Done' is selected in the *MODEL*^{ing} program
- the second set of questions depends on the answers given before so that only sensible and important questions for the specific situation are asked

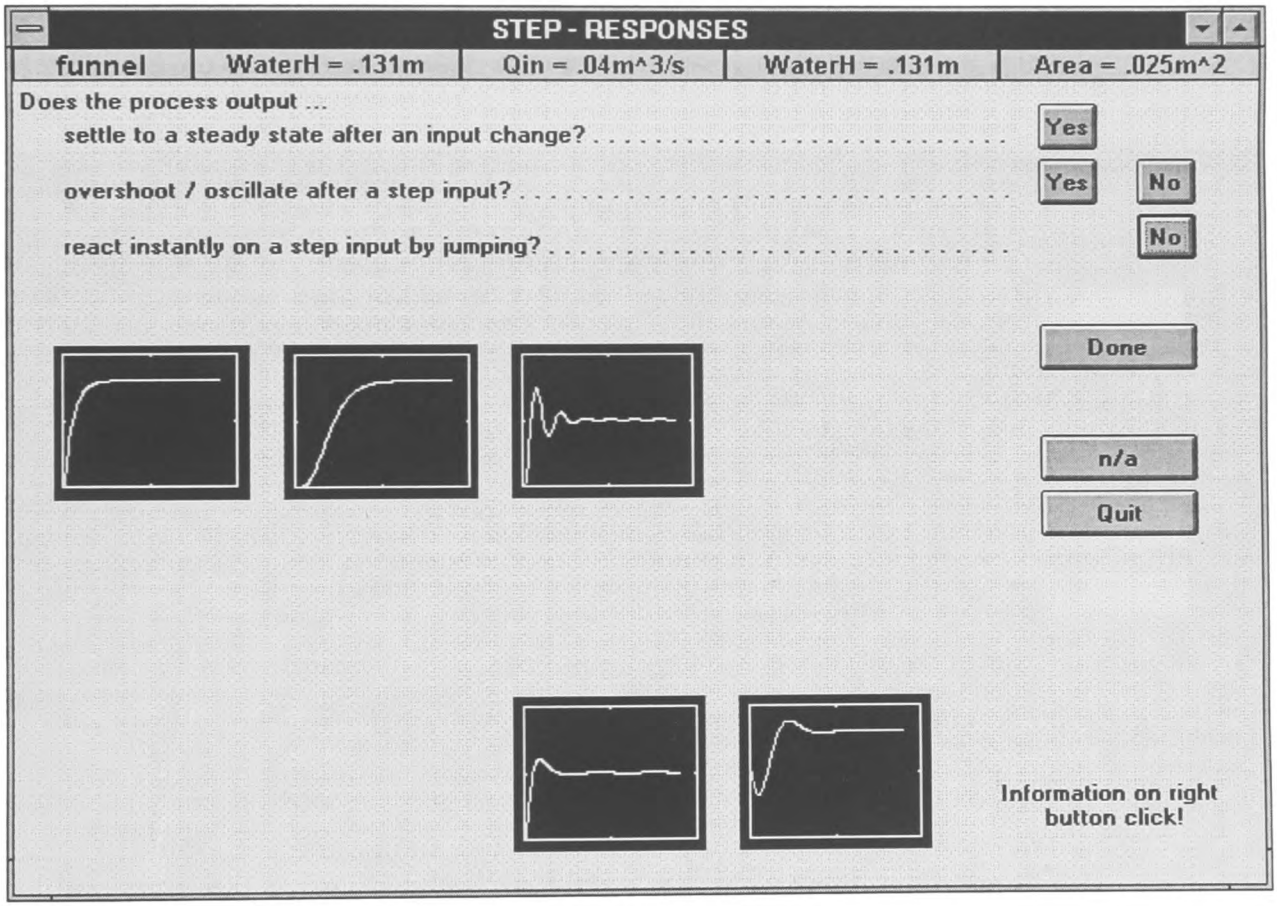


Figure 6-10: Step Responses With Decision Support Facility

To select one of the step responses directly necessitates implicitly the ability to answer several questions about the process simultaneously - a task which could be quite difficult for a non-expert in control engineering or modelling. Although it is normally possible to 'uncover' the implicit questions in the selection table of step responses with technical common sense, it could be faster - if not safer - to make use of the decision support in which the most important questions are explicitly

formulated. Having applied the decision support a couple of times, the user will probably be able to make a direct decision in the future (educational aspect of the decision support). Also, it could well be that it is impossible to answer all the implicit questions of the direct selection in a given situation. In such a situation, the decision support enables the user to enter the partial knowledge which is available. For further considerations on the decision support and its importance please refer to the discussion in Chapter 5.

Having selected an appropriate step response - either directly or applying the decision support - the user is requested to provide some numerical values of characteristic step response parameters in the 'Step Response Characteristics'-window, Figure 6-11, so that a parametrised model can be calculated. The values to be entered can either be rough estimates or derived from process data, like quality assurance notes or a quick, simple test of the process - the user is advised accordingly in a brief message. Again, the status bar is an important orientation aid in this situation.

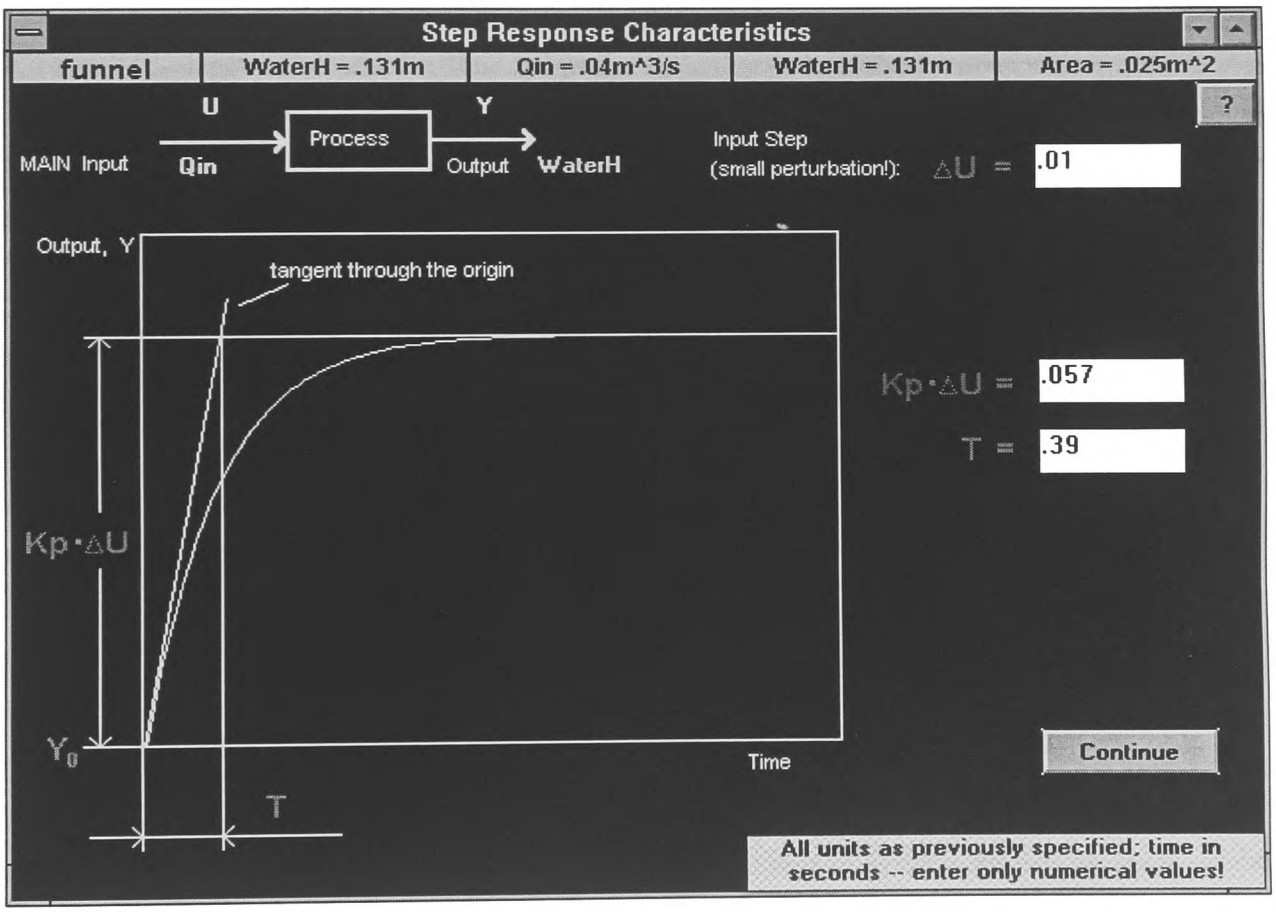


Figure 6-11: Step Response Characteristics

On the basis of the numerical information, the system determines automatically all relevant transfer function parameters, using simple parameter identification approaches (cf. Appendix A6). The linear

dynamic *MODEL*^{ing} sequence moves on to the 'Local Modelling Results' screen, where the inference result, i.e., the locally valid process model is displayed in control engineering terms (Figure 6-12):

- the transfer function structure
- the parameter values of the transfer function
- the model type shorthand (P - proportional, D - derivative action, I - integral action, Tx - x^{th} order lag)

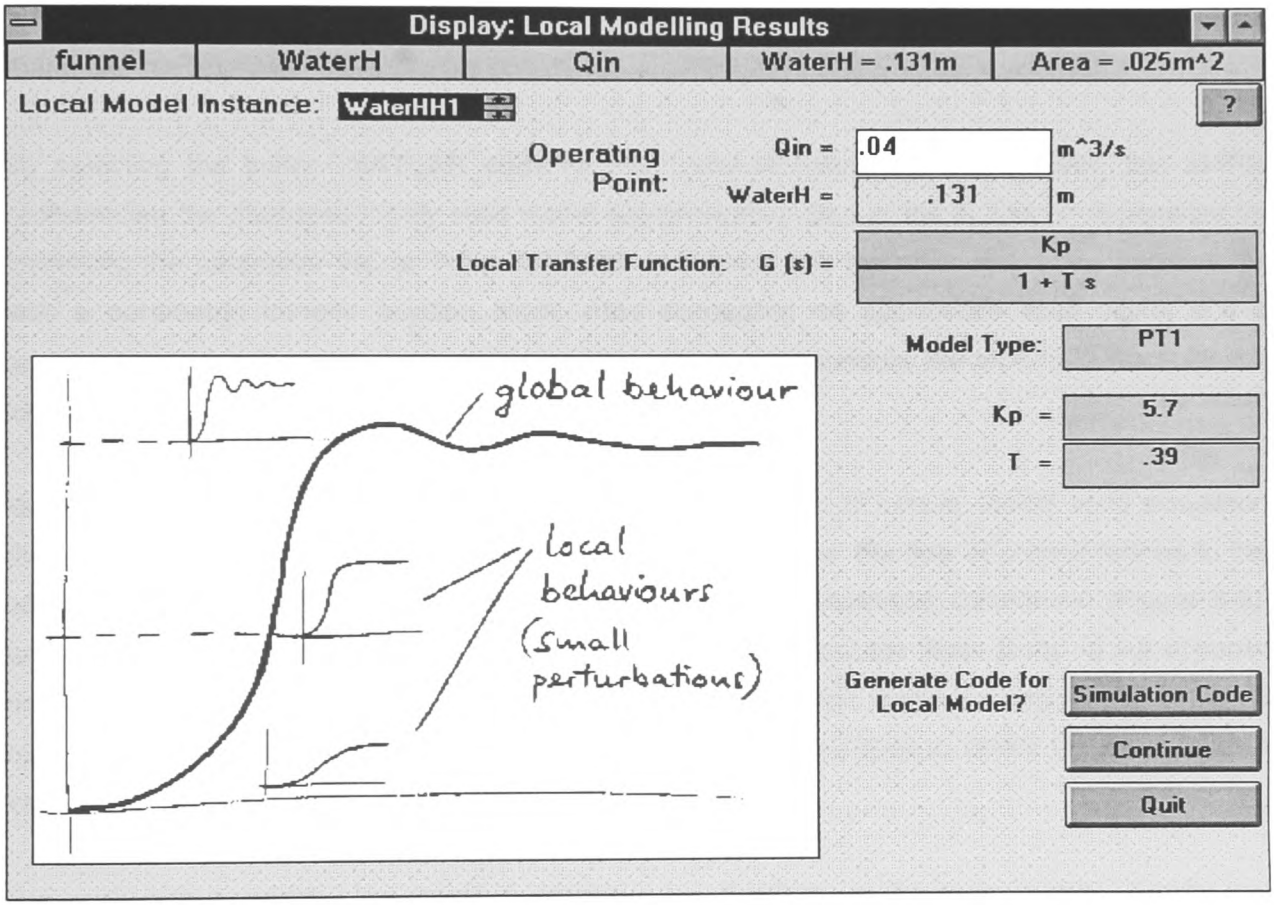


Figure 6-12: Local Modelling Results

Of particular importance is here the structural information which is deduced from the selected step response type. Even if some parameter information should remain incomplete, the structure will be particularly valuable for the following CACSD module on simplified experimental identification which is being developed in the collaborative research project.

The considered example of the funnel tank shows locally only first order proportional behaviour. This is in terms of parameter identification on the basis of the specified characteristic step response

values obviously one of the very trivial cases. The example that was published in [107] shows, however, how the built-in parameter identification becomes much more useful if the process is, for example, of sixth order. *MODEL^{ing}* and its prototype implementation are, nevertheless, not mainly about a particular parameter identification algorithm but rather about a new principle of simplified modelling for highly complex nonlinear multivariable processes. The very nonlinear and multivariable funnel tank is therefore still a very good demonstration example. In fact, the particular choice of identification equations and look-up tables that have been programmed (Appendix A6) is independent of the presented approach as such and should be complemented by other simple procedures that are, for example, based on the specification of the times that the process output takes to rise to certain percentages of its steady state level. These and other sensible extensions that make the approach more flexible and usable are discussed briefly in the last Chapter.

By selecting the button '**MATLAB code**' in the 'Local Modelling Results' window, two M-files representing the displayed locally valid model are generated. One of the M-files is responsible for initialising the parameter values in the MATLAB workspace and the other one is the Simulink file with a parametric transfer function block. After connecting an appropriate input signal and a graphical output in Simulink, a simulation can be carried out to validate the *MODEL^{ing}* result for the local operating point.

Since the main objective of the approach is not the generation of simple, locally valid simulation files, the user moves normally straight on by pressing **Continue**: the flow of control returns to the window with the heading 'Influences: Combination of Local Operating Conditions' (Figure 6-8), where the next combination of influences settings is selected and the 'Main Body' of the dynamic modelling (Fig. 5-9, 5-10) is re-run. This repetitive modelling cycle continues until all combinations of influences settings have been considered and the user is advised to select the **Next Output** variable.

For each output variable, the repetitive procedure is completed in the same fashion. Finally, the whole sequence returns to the 'Main Menu' (Figure 6-3). Using the functions in the menu bar of this window, the accumulated process knowledge (the global model) can be summarised in an ASCII Report file and MATLAB simulation files, whose features are discussed in the following two sections of this chapter. Via the **View** functions, these files can be checked within the *MODEL^{ing}* program. Figure 6-13 shows the M-file viewer for the funnel tank example. The program-internal, object-oriented representation of the component model can be viewed via the '**Object Tree**' option in the **View** menu (Figure 6-14). Double-clicking the instances in the object tree allows viewing their individual properties in form of slot entries. This particular viewing option gives good insights into the modelling concept but it is not meant for users with little modelling experience.

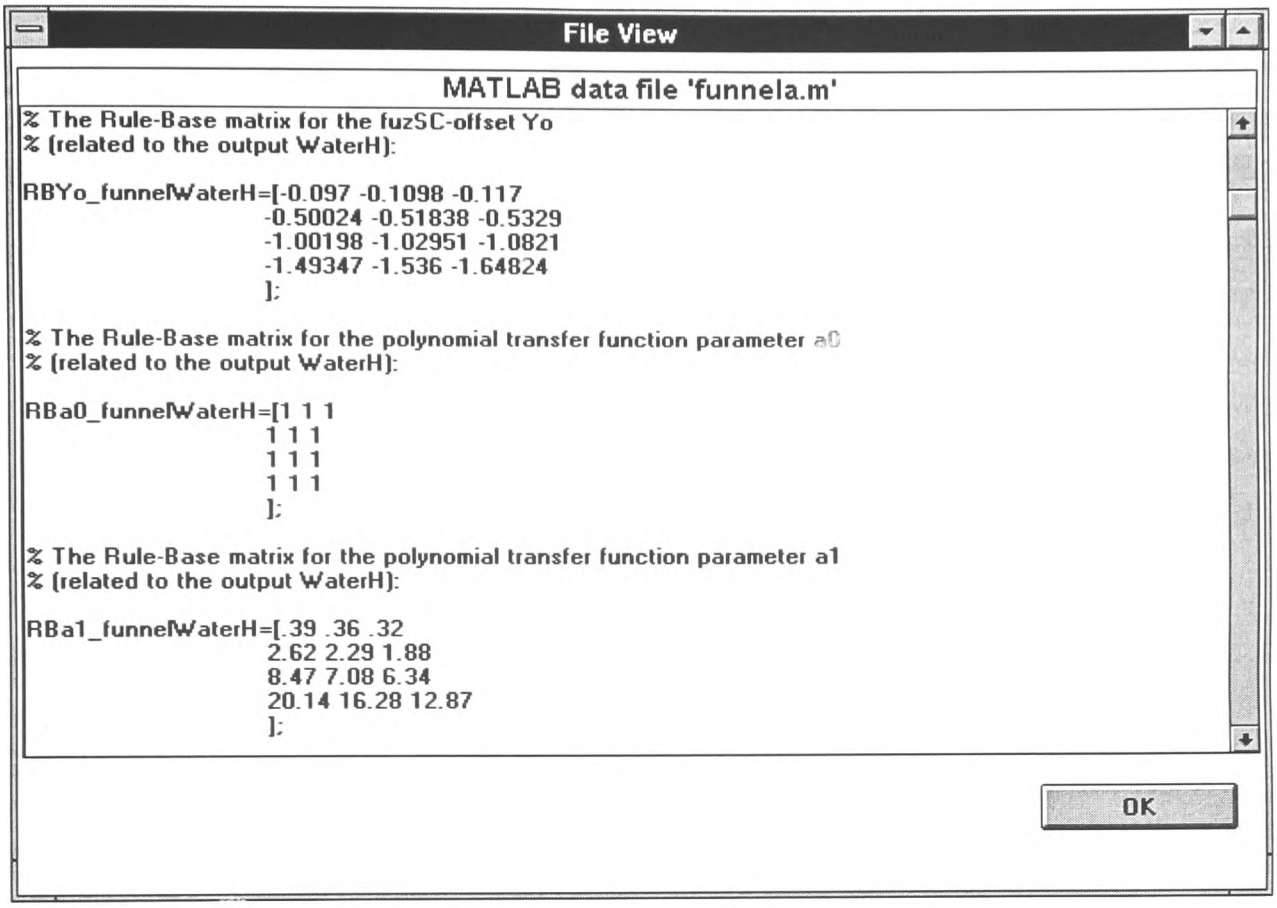


Figure 6-13: Viewing The Generated M-File

Following the presented procedure for our funnel tank example ensures the completion of the 'fuzTF' model in exactly the same format as it was shown in Chapter 4. Without the user realising - and, in fact, without him/her having to understand anything about "fuzzy" in general or "fuzTF" in particular - the model is correctly 'accumulated'; the parametric transfer function block (fuzTF), the static characteristic block (fuzSC), all parameter matrices, and the fuzzy membership functions are generated.

The currently implemented portion of the *MODEL*^{ing} sequence is thereby completed. As it was mentioned above, some of the less important intermediate states in the flow of control as well as the complete section on purely nonlinear *static* modelling had to be left out in this prototype implementation due to temporal constraints of the research project.

All remaining parts of the sequence should be implemented in the same user interface style as the current prototype. Especially the 'Generalisation and Specialisation' approach ('NLs4' in Fig. 5-12 and Fig. 5-15) should be graphically well supported with figures illustrating the requested

information in a similar fashion as the selection of step response types and the specification of step response characteristics.

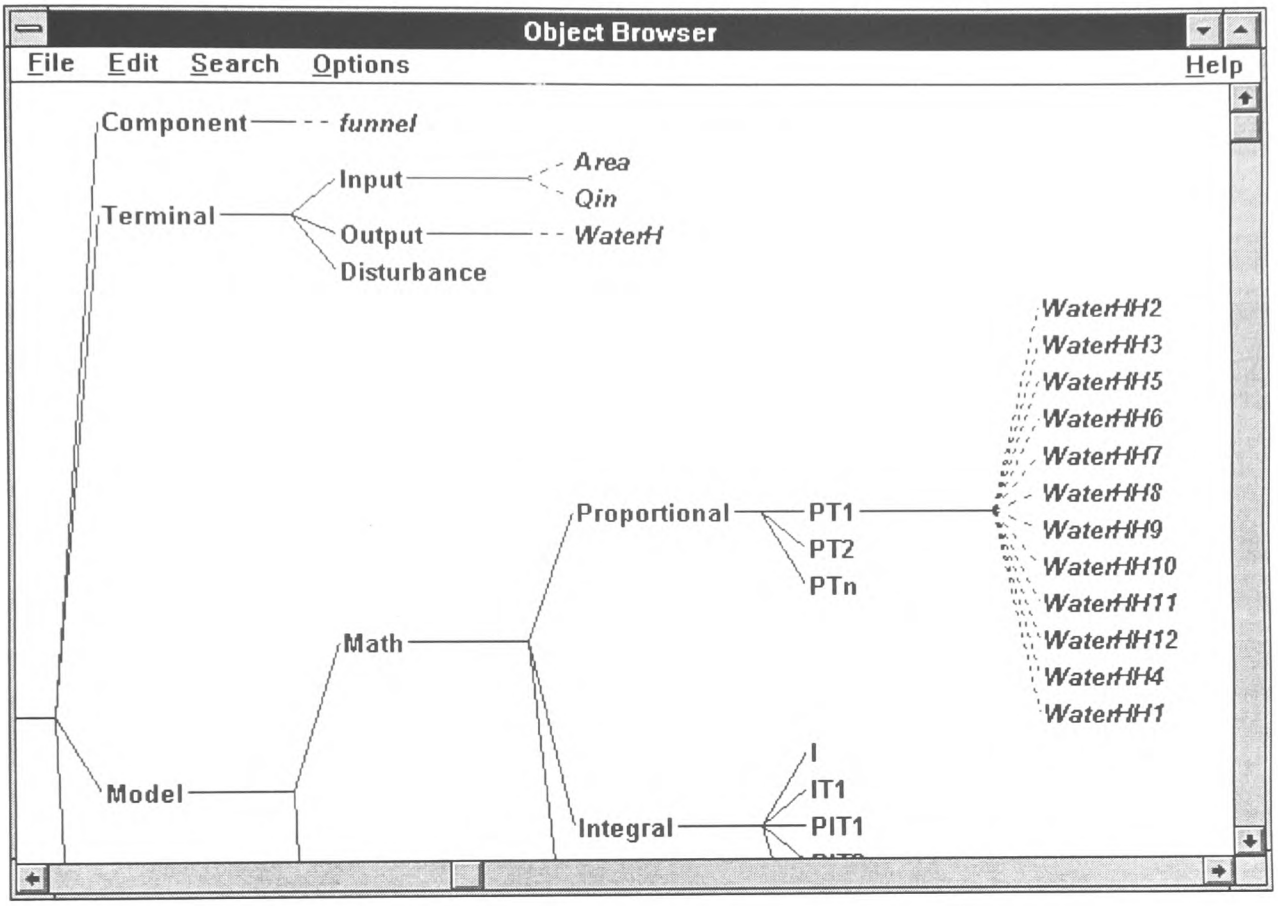


Figure 6-14: Viewing The Internal Model In Form Of Instances In The Object Tree

6.5 Summary Report Generator

The data flow diagram presented in Chapter 5 (Figure 5-16) illustrates the purpose of the summary report and thereby also demonstrates the importance of the report in putting the *MODEL*^{ing} approach into the context of its envisaged application: One of the central goals of the new modelling methodology was to facilitate the communication of the area engineer's process knowledge to the control experts. The summary report is the final 'medium' in this communication chain. Moreover, the report was also meant to give a feedback of the modelling process to the information provider him-/herself and to transfer some of the information directly to the other CACSD modules (Figure 5-16).

As in the previous section, the funnel tank is taken as an example. The above described modelling sequence results in the summary report[2] shown in Appendix A6. This report is structured into

- General Component Information
- Terminal Information
- Model Information
    - Dynamic Mathematical Descriptions
    - Nonlinear Static Characteristics
    - Production Rules
    - Attributes, Describing the Component (including some data associated to the attributes)

'Rules' and 'Attributes', the latter two sub-sections of the 'Model Information', are not yet acquired through the currently implemented modelling sequence. The data structure and the report generator, however, accommodate already most of the originally devised functionality. All data from the beginning of the ASCII-Report file for the funnel tank example in Appendix A6 up to the first set of nonlinear static characteristics (inclusively) has been acquired through the implemented approach. The remainder, commencing with the second set of static characteristics ("SC2_..."), refers to the purely static modelling approach and has therefore been manually input into the object tree in order to demonstrate the properties of the generated report.

Object orientation is not just a simplified means of systems design and programming but it is rooted in the human's perception of the world. The summary report is therefore not by chance structured according to the main classes of the *MODEL*^{ing} object model. According to its purpose as a means of communication between people, it has been specifically structured so that it is well understandable. Nevertheless, the layout of the generated report should be refined in order to further improve its readability. This could be achieved, for example, by arranging all data in standard columns and by highlighting particularly important information.

The ASCII-report is saved with the file extension ".m", so that it can be read by MATLAB. Typing the file name without extension (i.e., the component name with an appended "x", in our example: "funnelx") in the MATLAB workspace executes the file. Most of the lines in the report contain verbal information and would cause errors in MATLAB. These lines are therefore commented out using the "%"-sign so that only the numerical information is read into the workspace. In particular static characteristics that have been extracted in *MODEL*^{ing} from the nonlinear *dynamic* process information are transferred by this means in matrix-format to the simulation environment. Different

[2] the program-internal representation of the funnel model in form of a collection of instances with their slots is also shown for reference in Appendix A6

to the 'fuzSC' format, the static characteristics matrices in the report contain directly the input-output data (first column MAIN input settings, second column the associated output settings). Such two-dimensional static characteristics are only valid for specific settings of the influences. A collection of two-dimensional static characteristics is therefore required to fully describe the static behaviour of multivariable processes (cf. FUNNELX.M report in the Appendix A6).

In the first stage, probably only the numerator and denominator structure as well as the static characteristics information will be directly evaluated by the CACSD module that aims at an improved experimental process identification design. It is, however, also quite easily possible to search automatically for specific strings, like "Possibility of Impulse Input Signal", in the report file and to evaluate the following answer string ("Yes"/"No"/"NULL") in order to select the applicable input signal.

Another feature that improves the direct use of the derived transfer function structure is the file "identi.m", which is generated together with the ASCII-report. It is a Simulink file with the appropriate parametric transfer function (numerator and denominator of highest occurring order). This parametric transfer function block can be directly integrated into the experimental identification module.

In *MODEL*^{ing} generated ASCII-reports can vary significantly in length, depending on the complexity of the process which is, in particular, determined by the number of inputs and outputs. The report generator ensures this flexibility by checking systematically for existing instances and by adding information accordingly to the report. Some sections of the report are also conditionally left out: attributes details, for example, are only added to the report as appropriate (e.g. only further details on nonlinear static relationships if the question "Is the static relationship nonlinear?" can be answered with "Yes").

6.6 MATLAB Code Generator

As part of the repetitive modelling sequence, simulation code representing locally valid parts of the overall model can be generated. The application of this code for validation purposes was briefly described above (cf. also [107]). This section focuses therefore on the more important globally valid process model.

Similarly to the ASCII-report, the generated simulation code is responsible for transferring the accumulated process knowledge outside the *MODEL*^{ing} module and is therefore likewise important

(Chapter 5, Figure 5-16). The specific purpose of the code is the simulation of the *global*, nonlinear process behaviour (i.e., simulation over the whole operating range). Noteworthy about this aim is the fact that the model for global, nonlinear, multivariable simulation has been created from a 'patchwork' of locally valid, linear, singlevariable models which in turn have been acquired from some simple behavioural characteristics.

Again, the generated example files, named funnel.m, funnela.m and funnelb.m , are listed in Appendix A6. The code is split into the three files as follows:

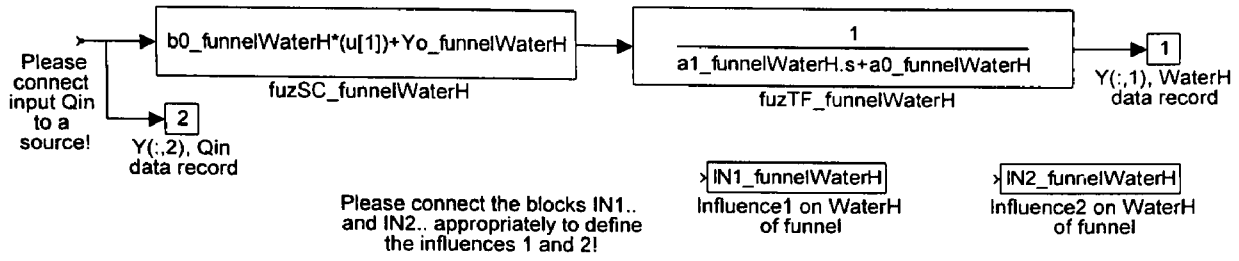
'<Component>.m' is the Simulink file with all 'fuzTF'-building blocks. For each output one fuzTF-block, one fuzSC-block and up to two ports for data transfer from Simulink to MATLAB workspace that define the nonlinear influences, as well as two further ports to record the MAIN input / output data. Figure 6-15 shows the blockdiagram for the funnel tank. Note that this example has only one output; multiple output process components are represented by several such blockdiagrams which are automatically arranged one after the other in the same window, which might have to be scrolled to view all parts of the model. The number of outputs handled and therefore the size of the overall blockdiagram is not limited by the implemented approach - only by practical aspects and the resources of the computer system. Before simulating the fuzTF-model using the purpose-built simulation function "fuztfsim.m" (cf. discussion in section 8.4), the influences ports must be connected appropriately and the input signals must be defined. The user is advised for these steps by messages in the blockdiagram window.

'<Component>a.m' is the data file which is responsible for defining all parameter matrices in the MATLAB workspace. It also triggers the execution of purpose-built functions that generate and display the fuzzy membership functions (Figure 6-16) and it passes the appropriate parameters to these functions. The data file is, similarly to the ASCII-report file, generated complete with explanations that define what each piece of data refers to. Please refer to these explanations in the funnela.m example to get a more detailed impression and compare the automatically generated rule base matrices with those presented in Chapter 4.

'<Component>b.m' is the Simulink file with the alternative static characteristic block that should be derived from the not yet implemented purely static modelling approach.

If the code generator comes across incomplete information, it advises the user on the parts that are still required. After the successful generation of simulation code, the user is advised on the application of the generated files. It should be noted, however, that the application of the code remains presently the task of users with some MATLAB experience.

a)



b)

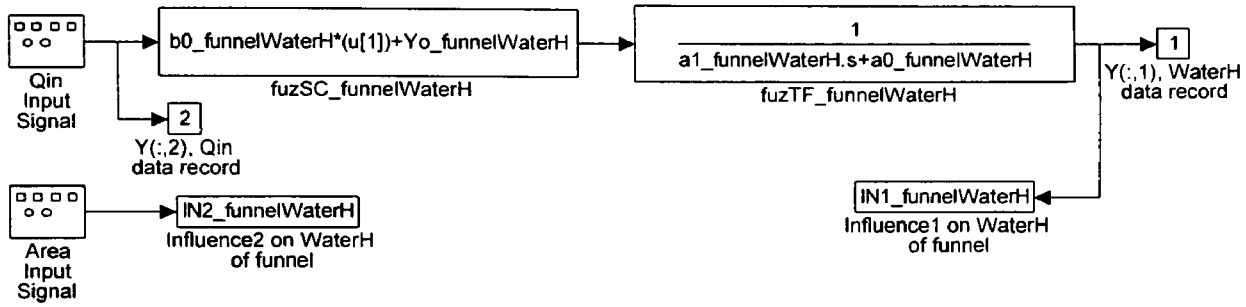


Figure 6-15: The Generated Blockdiagram For The Global Funnel Model

a) before, and b) after manually connecting the input and influence signals

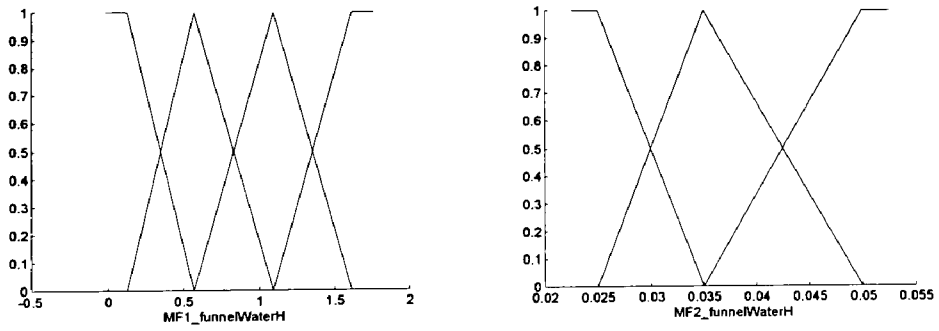


Figure 6-16: The Generated Membership Functions

Differently to the original plans, only a MATLAB code generator could be implemented by the end of the research project. The main purpose of the prototype implementation, however, was to show the feasibility of the conceptual design and thereby to validate it. After achieving this purpose, extensions to other simulation environments are now easy to imagine - yet they would involve a lot of routine work (cf. Further Work, Chapter 9.).

6.7 Chapter Summary

The *MODEL*^{ing} approach was implemented on PC using the object-oriented high level language KAL in the programming environment Kappa-PC. With about 12000 lines of KAL source code, the *MODEL*^{ing} approach is so far only partially implemented, but the program features most of the important multivariable nonlinear *dynamic* modelling, whereas the purely *static* modelling part remains to be implemented. About 140 functions, 50 methods (or 'demons') and only a mere 4 rules result in a mixture of mainly procedure and event driven software control.

To increase the robustness and usability of the program, various functions and methods have been programmed in order to catch erroneous or missing inputs and to post appropriate advice to the user. These functions should, however, be further extended to checking also for "sensibility" and consistency of the provided information.

Kappa-PC is very consistently object-oriented, which facilitated the implementation of the originally devised object model with only minor alterations. This property of the programming environment was also the basis of the realisation of the envisaged flexible frame-based knowledge representation. The required data handling and sorting procedures run in the background, hidden from the user, who can fully focus on the straightforward interactive behavioural modelling sequence.

Some of the aspects that have been discussed in this chapter are not only particular features of the prototype implementation but very important design considerations for the *MODEL*^{ing} approach as such. These aspects include, for example,

- the flexible knowledge representation
- a browser with systematic search support facility
- right button help on any parts of the user interface
- status bar for orientation
- advice on missing information
- different results viewing facilities
- "limited functionality without being restrictive"
- export facilities for the acquired model in form of simulation code and a 'plain English' report

Some experiences with the application of the *MODEL*^{ing} prototype are summarised in the following chapter.

7. Validation Of The Approach

This chapter gives a brief summary of the testing carried out in order to verify and validate the proposed novel approach. "Approach", in this overall view, refers obviously to the knowledge elicitation methodology '*MODEL*^{ing}' and its GUI, which was proposed in Chapters 5 and 6, respectively. The *MODEL*^{ing} methodology, however, builds on the second key contribution of this work, the fuzzy hybrid modelling approach, 'fuzTF' (Chapter 4). A third novel approach is the attributes-based modelling, in particular of static characteristics. The latter method, which plays a somewhat less significant role in this work, was presented as part of *MODEL*^{ing} in Chapter 5. All three contributions are considered individually in each of the following sections of this Chapter.

7.1 Implementation Of The Prototypes

The implementations of both the fuzTF and the *MODEL*^{ing} approach in computer programs were important parts of the validation process since they required a thorough reconsideration of the logical consistency and algorithmic completeness of the proposals. Hence, they were a first, general test to demonstrate the feasibility of the novel approaches.

As part of the static modelling within the *MODEL*^{ing} approach, the novel method of modelling by collecting applicable descriptive attributes (proposed in Chapter 5), however, has not been implemented in a program and therefore not been tested. In any case, this concept as such does not require verification or validation because it is applied on a day-to-day basis by virtually everybody. What is new about this approach is the idea to *systematically* apply it as a means of efficiently *summarising* and *communicating relevant experiences* of process behaviours, so that either a human control expert or an appropriate computer program is able to derive a suitable experimental process identification set-up. This abstract (or 'basic') kind of knowledge could be particularly useful for the consideration of nonlinearities (e.g. hystereses) in the design of experiments and is being evaluated as part of one of the other research projects within the overall collaborative work.

With respect to the GUI, the process of implementation supported as much the design as it supported the testing - both tasks were carried out in a 'closed feedback loop' throughout the implementation phase.

7.2 Validation Tests Of The Implemented Work

The fuzzy hybrid approach 'fuzTF' was initially tested by reproducing extremely nonlinear, hypothetical behaviours of processes whose locally valid, linearised transfer functions would differ significantly with respect to the order of magnitude of their parameters and even in their order. A first test criterion for such simulations without physical reference was the visual check whether the generated global simulation results were intuitively valid. Secondly, local trajectories were generated under incrementally altered conditions of the nonlinear influence parameters. These tests showed successfully the correct 'intermediate' dynamic responses between previously known operating conditions. An example of this kind of testing illustrated the introduction of the 'fuzTF' approach in Chapter 4 (cf. Figure 4-8). Furthermore, intermediate[1] trajectories that had been generated through the fuzTF-model were compared with those generated with the well-established Takagi-Sugeno model (Figure 4-10). The results of these latter comparisons supported the analytically derived conclusion that the interpolation of dynamic trajectories (Takagi-Sugeno) is, as opposed to parameter interpolation and adaptation (fuzTF), not a sound technique to generate 'intermediate behaviours' between previously known trajectories.

In order to get a clear picture of the interpolation properties of the fuzzy static characteristic 'fuzSC' (which is part of the proposed global fuzzy hybrid model), it was applied to a set of benchmark problems (cf. Appendix A4) and compared with other interpolations. These problems included all challenging conditions that can typically appear in real situations. The results were generally very satisfactory and underpinned the flexibility of the fuzSC approach, which even addresses multivariability in its standard concept (full details in Appendix A4). In the overall context of this work, the flexibility of fuzSC was an important basis for the development of a standard approach to handling information on static process characteristics.

In the 'funnel tank' example, the global fuzTF model was then specifically tested against a theoretically derived nonlinear mathematical model, which represented the 'ideal' behaviour of the process (Application Example in Chapter 4). Neither the particular properties of the funnel tank, which made it an ideal test case for these validation experiments, nor the detailed results of the tests are repeated here. The general outcome of the tests, however, can be summarised as being very satisfactory.

[1] between pre-specified local behaviours

After implementing the *MODEL*^{ing} approach, the knowledge acquisition program was used to automatically generate the fuzTF model of the funnel tank example. The result was identical with the model that had been 'manually' built for the earlier tests of 'fuzTF' as such, which was a first successful test to show the validity and algorithmic correctness of the modelling procedure and its implementation (cf. 'manual' modelling in section 4.4 and the automatically generated M-file in Appendix A6).

Similarly to the fuzTF approach, *MODEL*^{ing} was additionally applied to hypothetical modelling situations. Modelling processes without physical manifestation was, again, necessary to test the more 'extreme' capabilities of the approach since appropriate real process information was not available. These tests focused on trying all features of the currently implemented status of the program - in particular the knowledge acquisition, simulation code generation and ASCII report generation for very complex modelling situations with various input and output variables. The criterion for these function tests was the correct handling of the provided knowledge according to the conceptual design. Each piece of input information was compared with its program-internal representation, i.e., its correct allocation to particular slots in the object structure. Likewise, the information was tracked to the generated simulation code and the ASCII report. Another important aspect was the program's stability in case of incomplete information.

It was intended to test the approach ultimately in an industrial environment by firstly acquiring locally valid information on the dynamics of a process, generating a simulation model and carrying out a series of nonlinear ('global') process simulations, which finally should have been compared with the equivalent tests of the real process. These tests, however, could not be carried out, mainly because they would have required a significant involvement of industrial employees and a process which allows for the 'global' testing throughout the operating range to validate the simulation results. As long as the *MODEL*^{ing} approach is a standalone tool, the partners in industry would not benefit from such testing. Hence, the envisaged tests can only be carried out once further CACSD modules that build specifically on the proposed global simulation model will be available. At that stage, the collaborative research group could offer the complete 'service' of process and control system analysis, simulation, design and optimisation, so that a mutual benefit of the testing could be guaranteed.

In order to test the approach in a laboratory setting on a small scale process, a final year project was set up. This project involved the design of a water tank in which different perspex modules could be inserted in order to create various nonlinearly shaped tank volumes - resulting in different nonlinear outflow behaviours. Unfortunately, this project has been delayed so that no laboratory tests could be carried out to date.

Obviously, it would have been nice to round off this project with a "real" application. Nevertheless, such testing would proceed in exactly the same way as the testing which is based on a theoretically defined and locally simulated process. Furthermore, tests with a real process could never provide as much and as detailed feedback on the work as the simulation-based validation carried out with the funnel tank. One of the reasons for this is the fact that the simulation is free from unpredictable side-effects. Such side-effects, which appear in almost every real process, cannot be modelled; they would therefore falsify any statement about the quality of the modelling approach. A trivial example for such side-effects are algae, which can affect unsystematically the outflow in a laboratory tank, making experiments not well reproducible.

Hence, the requirements for the validation of a *modelling approach* are quite different from those for the validation of an *individual model* of a particular real world process. Whereas the question to be answered in the latter situation is "*How well does the model represent the overall behaviour of the real process?*" (this 'overall behaviour' includes possible side-effects), the question in the former situation is "*How well can a defined behaviour be reproduced by the new model type?*". In the validation of an *individual model* of a particular real world process it is therefore the purpose to consider all naturally occurring side-effects (which should be negligible - otherwise the validation result is negative). In this case, a negative validation result could be due to insufficient refinement of the model ***or*** it could be due to an inappropriate modelling approach ***or*** it could be due to excessive side-effects in the process. "Excessive side-effects" could refer in extreme cases to a "chaotic" property of the process which cannot be modelled at all on the basis of a finite set of information or data (example: wastewater treatment plant). In the validation of a *modelling approach*, on the other hand, all side-effects ought to be excluded so that after adequate model refinement any statement about the applicability of the model refers unambiguously to the modelling approach as such.

Another advantage of the purely simulation-based validation of the modelling approach is the facility to apply more 'extreme' test signals as well as the certainty that the input test signals are exactly the same. Overall, the concurrent simulation of the 'real' process and its model ensures clearly defined side-conditions and comparability of the results for 'ideal' and 'approximated' behaviour.

7.3 Test Of The Knowledge Acquisition Approach By Uninitiated Users

More than in any of the above kinds of tests, the application of the *MODEL^{ing}* prototype by uninitiated users gave a feedback on the quality of the GUI as well as an overall impression of the program.

Experiences gathered with the help of the first 'test-users' showed that some difficulties in understanding particular requests resulted from the unawareness of the underlying 'fuzTF'-modelling concept. As a reaction to this feedback, the explanatory free-hand sketches (cf. figures in Chapter 6) were added to the interface. Without giving any unnecessary details about the 'fuzTF' approach, the illustrations help the user to understand the general point of view that is taken in the course of modelling. The response to these additions was very positive - in particular, because the free-hand sketches are unmistakably general advice rather than related to the particular problem at hand. Additionally, some of the requests were re-formulated or presented in an improved format and some input facilities were amended. Formal improvements of the interface were, however, sometimes limited by the prototyping facilities in Kappa-PC.

The initial tests also showed, that the 'Right-Button-Help' function, which is available in all windows of the modelling sequence, was not used at all. The fact that the testers had anyway only few "hick-ups" in the handling of the program indicated that the aim to develop a self-explanatory interface has largely been achieved.

A small message at the start of the program points the user now specifically to the 'Right-Button-Help' facility, which is particularly aimed to provide easily accessible advice to first-time users. Interestingly, however, this advice was still not followed by users with a control engineering background who tend to forget about this note after reading it and try to find their way through the program themselves as they are used to from other applications. Users inexperienced in control engineering, on the other hand, made extensive use of the help facility - they generally appeared to be more readily following the knowledge acquisition sequence and guidance than experienced users. In fact, a more differentiated and extended 'Right-Button-Help' facility was considered one of the major potential improvements of the implementation.

Another particularly interesting suggestion to improve the handling of the program for users with little control engineering background is the addition of an on-line glossary which should explain all terminology used. Finally, it was suggested to add to the brief information on each modelling step a likewise brief preview of what the following step is about - and how it is affected by the information given in the current modelling step.

Obviously, control-experienced users did not make use of the decision support facility for the selection of an applicable step response. Users without a control background, however, appreciated this facility very much.

As a whole, the feedback on the GUI was very positive - especially after the implementation of various small improvements. The current status is, however, still at a prototype stage, which has some further scope for minor refinements before it can be considered as fully self-explanatory for uninitiated industrial users. The main keys to the required refinements are the above stated comments and suggestions by uninitiated users. Likewise, error handling and data type checking has only been implemented to a limited extent in the current prototype.

All feedback on the overall impression of the knowledge based modelling program was also positive - in particular the facility to generate nonlinear MIMO models on the basis of such simple information was generally very much appreciated. The main reason for the easy handling and the good guidance was seen in the structured modelling sequence, which should not be altered but only further supported by the above mentioned additions to the GUI and the help functions.

Some concern, however, was expressed about the amount of requested information and the modelling effort involved. These comments resulted largely from the fact that users felt encouraged to make use of the high degrees of freedom and to specify as many influences and operating conditions as possible. Hence, the 'combinatorial explosion' effect led to overly complex modelling situations. As a first reaction to this feedback, some notes that advise to start with limited complexity have been added to the interface. Since complexity and practicality aspects have been taken very seriously, this issue is taken up again and discussed in more detail in the following Chapter, which gives a complete review and discussion of this research project.

8. General Discussion

This Chapter gives a brief review of the research *methodology* employed in the different stages of this work, from the broad starting point to a specific definition of this particular project as part of the collaboration, and further on throughout the project up to its completion. Furthermore, all major design decisions and results are re-considered and discussed on the basis of the experiences gained. The levels of fulfilment with respect to the original goals are also discussed where applicable and a résumé of the overall results in comparison with the expectations is given in section 8.5.

8.1 Decision On The Aim Of The Project

As part of the collaborative research project between the University of Glamorgan and the Fachhochschule Hannover, this research project started off on the basis of a very broad overall problem statement[1] "None of the existing CACSD programs addresses the need of area engineers in industry".

In order to get a clearer picture of sensible emphases for this project, a variety of methodologies was applied. Of particular importance in this respect was the 'hands-on' overview of existing CACSD programs and the studies of literature on CACSD developments, which indicated especially shortcomings in the area of process modelling. Following this finding, the familiarisation with theoretical and experimental mathematical modelling approaches and the review and analysis of library-based modelling approaches led finally to the conclusion that the process information, which is most commonly available among 'area engineers' in industry, is inapplicable to any of the existing approaches. Hence, the problem statement was refined as:

THE PROBLEM

- *Generally, the current Computer Aided Control System Design (CACSD) programs are not geared to the requirements and skill level of area engineers in industry.*
- *In particular, there is not yet an approach existing which allows engineers - except for control or modelling experts - to build a process model suitable for systematic control system design.*

[1] This starting point was based on Professor Schumann's experience as chairman of the VDI-Workshops 'Regelungstechnische Programmpakete' ('Control Engineering Programs'); 1989, 1991 and 1993 in Düsseldorf and his reviews of this field [125, 126, 127].

THE NEED

Rather than being built on theoretical systems analysis, a practical process modelling approach must be based on the experience with respect to process behaviour that engineers in industry accumulate during their everyday work.

On this basis, the decision to pursue the "Knowledge Engineering Approach to Process Modelling" was made. The detailed subject, need and aim of this work have been described in Chapter 1.

From the present point of view, after completing the project, all considerations that led to the decision for this emphasis of the work are still as valid as were some years ago, since the specific problem has not been dealt with in the literature.

8.2 Research Into Modelling On The Basis Of Partial Knowledge

The idea of the "Knowledge Engineering Approach to Process Modelling" included the concept of making full use of the available process knowledge of different types. "Available" is here the key to the consideration that it may be difficult to fully define conventional process models. This led to a thorough investigation into modelling approaches that deal with some kind of 'qualitative', i.e., incomplete or partial, information (Chapter 3).

The conclusion of the qualitative modelling review led to the further focus on the fuzzy modelling approach. This was always likely to lead to controversial views among researchers in the field of qualitative modelling. In fact, fuzzy modelling as such is not generally acknowledged as being one of the qualitative modelling approaches, which is largely due to the fact that no standard definition of this field exists. It is therefore all too easy to exclude fuzzy modelling - a state of affairs that is also strongly criticised by Sugeno and Yasukawa [94]. Hence, the decision to use fuzzy modelling can still be justified; in particular in the context of this work. Nevertheless, it was pointed out in Chapter 3, that another qualitative approach should be considered additionally in order to cover qualitative aspects more fully (cf. 'Further Work' in Chapter 9).

Starting from this decision to use fuzzy modelling and the criticism of its limitations with respect to modelling dynamic properties, the development of the novel fuzzy hybrid approach commenced. Part of this development was again the consideration of previous, related work in order to determine the exact point to progress from. Concurrently with the development, the new concepts were checked for logical consistency and implemented in MATLAB for validation. The proposed default

settings (membership functions, etc.) were also tested and selected among a variety of possibilities with the help of this prototype implementation.

Overall, the results of the development and testing of the 'fuzTF' approach turned out to be much better than could be foreseen: what used to be considered with some scepticism about the likelihood of success as "the icing on the cake" (i.e., on this project) amongst the members of the collaborative research group, emerged as the first major contribution of this work. Whilst the novelty of this approach may not at first sight be apparent since it is built on well known and tested techniques (fuzzy logic, parameter adaptation, etc.), the concept of combining those techniques and applying them in this particular manner for the purpose of process modelling and simulation is entirely new and it only emerged after a series of unsuccessful concepts had been developed and scrapped. The impact of these positive results on the overall work is discussed in the following sections.

8.3 Research And Development Of The Knowledge Acquisition Procedure

The original aim of the knowledge engineering approach was to systematically elicit as much of the potentially available process knowledge as possible. This included in particular the idea to address all *types* of knowledge (Appendix A2), with an emphasis on the most 'fruitful' types, *heuristic* and *causal* knowledge. Due to its specific properties (cf. A2), *causal* knowledge is traditionally the type of knowledge most commonly applied in process modelling. *Heuristic* knowledge, however, is after *causal* knowledge the second most important knowledge type for modelling (cf. discussion in Chapter 5) and, as opposed to *causal* knowledge based modelling approaches, hardly developed on a systematic basis.

Due to the temporal constraints of this project, the extensive work on qualitative approaches and the development of the 'fuzTF' approach, the work had to focus solely on the acquisition and use of *heuristic* knowledge. *Heuristic* knowledge is, however, not only of major interest because of the limited attention it has received in the field of process modelling. Of particular importance for this work is the fact that this type of process knowledge is most likely to be available among area engineers in industry.

The successful development and test of the 'fuzTF' approach led to the conclusion that the knowledge acquisition procedure should be designed along the lines of the conceptual systems view of the proposed fuzzy hybrid model. This means specifically that in the course of modelling nonlinear multivariable dynamic properties of processes, the knowledge elicitation focuses at local

operating conditions and aims to derive piecewise linear transfer functions. Hence, 'fuzTF' formed the basis of breaking down the task of MIMO modelling into repetitive MISO modelling, which in turn is split into SISO modelling sequences (Figure 5-5). Therefore, the consideration of the 'fuzTF' approach as an integral part of the knowledge acquisition methodology proved to be essential in the simplified handling of highly complex processes.

Another key influence on the knowledge elicitation procedure both with respect to its result and with respect to the process of its development was the notion of Object-Orientation. It helped to analyse and structure the information to be acquired and thereby formed the basis of the knowledge representation format, which is at the same time the simulation-independent model representation. The object-oriented approach further supported the design and documentation of the sequential flow of control in the knowledge acquisition. Finally, this notion facilitated the direct implementation of the conceptual design into a - likewise object-oriented - prototype program. Amongst the many advantages of object-orientation, these are just some of the specific benefits enjoyed in this work.

The design of the sequential flow of control was a particularly important part of the work on the knowledge acquisition approach. It was repeatedly revised and improved in order to optimise the efficiency (i.e., the effort of its later application) and its self-explanatory character.

A very important early design decision was to ensure generality in as many aspects as possible. This included in particular the applicability to processes from any engineering domain and to components of any 'granularity' - thus, dealing with anything from an elementary sub-component to a complex plant. The applicability of the developed approach depends only on the availability of the required kind of behavioural process information, which makes the knowledge engineering approach even usable outside the engineering domains. An inevitable side-effect of the required generality is the significant amount of requested information, which is not always applicable to the particular modelling situation. In an attempt to reduce this drawback, the most essential information should be characteristically highlighted in the user interface of the modelling program (cf. following section).

8.4 Implementation

The Prototype Implementation Of The 'Fuzzy Hybrid' Simulation Facility:

MATLAB was used to implement, try and test the proposed fuzzy hybrid modelling approach - mainly because of its quite simple and flexible extendibility via functions in ASCII format. However,

MATLAB is (at least until version 4.2) not built to accommodate systems whose dynamic transfer function parameters vary during the simulation, which is exactly what was needed for the implementation of the 'fuzTF' approach. The only possibility is to read and write new parameters from and to workspace at the beginning and end of a simulation, respectively. As a way around the MATLAB limitation, a "script-file" was therefore written which triggers the continuous simulation run for a small time increment, updates the parameters and re-starts the incremental simulation in a repetitive fashion. This piecewise - or incremental - continuous simulation is therefore not part of the fuzzy-hybrid modelling approach as such, but only a way around the limitations of the implementation environment.

In order to proceed at the beginning of each new increment with the final system states of the preceding simulation increment, the state matrix is passed on. The peculiar MATLAB feature of including the highest order denominator coefficient of transfer functions as factors in the state matrix required the normalisation of those states that are associated with the fuzzy adapted transfer functions in this matrix. After the normalisation, the appropriate matrix positions are multiplied with the updated coefficients.

As opposed to these procedures, the process of updating the transfer function parameters as such is a standard part of the proposed 'fuzTF' approach. It consists of the fuzzification of influences, the inference, and defuzzification, which are combined in an efficient matrix operation (cf. Chapter 4).

The implementation of the simulation script-file ("fuztfsim.m") turned out to be relatively complex, particularly due to the programmatic effort to facilitate the handling of the user-defined component name(s) and the individual parameters of each component, which all have to be adapted appropriately.

Using "fuztfsim.m", the incremental simulation step sizes can be either user-defined or automatically adjusted. Although the interactive sequence makes the handling of fuztfsim.m fairly straight forward, it is important to follow the instructions at the start carefully and to enter the information exactly as required. It is important to note that, despite the attempts to simplify the use of fuztfsim.m, it is only a first prototype for validation purposes of the fuzzy hybrid ('fuzTF') approach, which is - as opposed to the *MODEL*^{ing} sequence - currently not geared at uninitiated users. Apart from the 'handling', there is great potential for improvements with respect to the execution time of fuztfsim.m, which could be achieved through more efficient algorithms and an implementation in C-code.

The limitations of the 'fuzTF' test-implementation are:

- The global nonlinear models can only be combined from locally valid *proportional* models of any order or from transfer functions with a factor linear in all numerator parameters.
- A maximum of two nonlinear influences can be considered.
- A maximum of 9 levels (i.e., typical operating points) per influence.
- Conventional nonlinear elements (like dead time or limiters) with varying parameters cannot be simulated.

Despite these limitations of the prototype implementation, the proposed approach as such is not limited with respect to any specific transfer function structure or the number of influences and their settings. Even nonlinear elements with varying parameters (e.g. varying dead times) can be handled via the proposed approach, which opens up an extraordinary wide range of process characteristics that can be covered in a single approach. MATLAB 4.2, however, does not provide the ideal simulation environment to take full advantage of the strengths of the 'fuzTF' approach. The new version, MATLAB 5, appears to overcome some of the key limitations: in particular the extension from 2- to n-dimensional matrices facilitates the consideration of any number of influences per process output.

The Programming Environment For The *MODEL*^{ing} Prototype:

Kappa PC was chosen as the programming environment for the prototype implementation of the *MODEL*^{ing} approach. The main considerations for this choice were its strong object orientation, that it is PC-based, its suitability for rapid prototyping purposes, and finally, its flexibility with respect to the software control paradigm.

In the course of implementing the knowledge acquisition procedure, some problems and limitations of Kappa PC led to hold-ups and substantial effort to work around them. Among the problems and limitations are:

- bugs (e.g. in the earlier release 2.0.11: GUI elements which change position during the development)
- no local variables
- no pointers / call by reference
- variables only as slots of the types: text, number, boolean, object, with the option single or multiple valued (i.e. lists of values), but no matrices.
- problems with maths: e.g. *"0.3"* and *".3"* can both be used for calculations but are not considered equal in comparisons
- limited function sizes

- associations between objects cannot be programmed properly according the object oriented notion

In particular the latter point is quite disappointing for a highly object oriented programming environment. Although one might expect the appropriate functionality behind the slot type 'object', its handling and properties are, in fact, not different to those of an ordinary 'text' slot. The concept of several instances 'belonging' together, forming the flexible frame based knowledge representation which was proposed in Chapter 5, would be ideally represented by associations. Due to the absence of such a formalism in Kappa PC, the appropriate data handling and access functions had to be programmed manually.

In view of these problems, a different programming environment might be preferable, although the considerations that led to the selection of Kappa PC as a prototyping tool are still very good arguments. For the final implementation of the approach, however, the selection criteria will be differently emphasised so that Kappa PC will almost certainly not be the first choice. A thorough analysis of Object-Oriented programming languages should precede such a final selection, and it is certainly a good idea to consider especially C++ in some detail.

The Data Structure:

The designed object model was directly implemented with only minor changes in the *MODEL*^{ing} prototype, where it formed the basis of the system's internal data structure. The proposed flexible knowledge representation format in the form of a flexible frame was implemented via special slots that keep track of the associated instances and via purpose-built sorting and access functions (Section 6.3); it works very well and fulfils the expectations.

In relation to the final model as it is exported for simulation purposes, the program-internal, object-oriented representation of the component model is an 'abstract' intermediate model, which is neutral in the sense that it is independent of any particular simulation program.

The Interface Design:

Essentially, the main consideration in the interface design was how to express very complex things in a few clear and simple words and how to support the information requests and user inputs visually.

The fuzTF concept helped considerably to address this highly complex issue of multivariable nonlinear modelling in a simplified fashion. In particular, it allowed breaking down the complex task into simple, local 'units', which proved to be essential. Still though, it was sometimes difficult to formulate information requests and to design the appropriate interface in such a way that the user is not confused. With the introduction of explanatory free-hand sketches, the user can get an idea of the general point of view that is taken in the course of modelling, which helps substantially to understand some of the information requests. Additionally, the wording in the requests for user input has been improved iteratively with the help of test-users. Further refinements might, however, be necessary. These refinements ought to be carried out in collaboration with as many potential users as possible, in order to find out, where potential sources of misunderstandings are hidden.

Overall, the problem of 'how to express complex things in simple words' had been underestimated. The main reason for this problem lies in the varying backgrounds of potential users. What user "A" understands easily could confuse user "B", for example. Nevertheless, the simplified knowledge acquisition structure ensures that the remaining complexity can be handled - at least when the mentioned 'fine-tuning' of the wording will be completed - in a truly self-explanatory manner. Another attractive way of addressing users of all backgrounds is to provide a comprehensive on-line glossary, as it was mentioned in Chapter 7. A direct access to this glossary via hypertext (i.e., by clicking on the word in question) would be ideal.

The Issue "Modelling Effort":

As discussed in Chapter 7, the concern about the amount of requested information, which was expressed by some of the test users, resulted largely from the risk of misusing the high degrees of freedom in the modelling approach. A first fix of this problem was introduced by adding the advice to start off with a limited number of influences and influence settings and to increase the complexity only as needed. This advice is shown at different stages in the course of modelling.

Although the degrees of freedom and therefore the modelling effort can largely be controlled by advising the user, a certain conflict between the two major goals of the work inevitably remains:

1. To minimise the effort of the user in the application of CACSD as a whole to yield maximum results.
2. To acquire as much of the available and potentially useful process knowledge as possible in the particular CACSD module developed in this work.

Knowledge, however, has to be acquired interactively and the more knowledge to be acquired, the more interaction, i.e., the more effort, too. This relation cannot be overcome, but only optimised. Minimising the user's effort was always a central focus of this work, as it has been pointed out repeatedly throughout the development of the knowledge acquisition procedure. Nevertheless, there are some important ways forward, which should be considered in further extensions of this work.

- The development of algorithms that handle incomplete parameter matrices within the fuzTF approach could decouple to some extent the direct relation between the user-specified degrees of freedom and the modelling effort. A relatively simple way of achieving this is the interpolation of the parameters at missing matrix positions. Obviously, this would have a negative effect on the quality of the process model.
- A data interface to the process or to a supervisory control system could facilitate the direct use of 'technical data', like operating ranges, tolerances etc. and thereby reduce the modelling effort for the user. Such a facility, however, should be no more than an extra feature of this approach rather than a main path to rely on; after all, this work on knowledge acquisition for process modelling addresses in particular those situations in industry where there is no computer-integrated supervisory control system but only basic, conventional equipment. Another important consideration in this respect is the fact that other approaches might be more suitable if process data is directly accessible for the program. The assumption at the beginning of this work excluded this area, which has been and is being dealt with in many other research projects.
- A very simple way of reducing the modelling effort is to distinguish more clearly in the GUI between essential information and additional details, which can be left blank. Furthermore, the warning and advice feature, which has already been considered in the design of the *MODEL*^{ing} procedure (Chapter 5), remains to be implemented. This advice feature is aimed to support a systematic complexity reduction.
- Finally, it might be sensible to give the user the choice between different knowledge acquisition emphases, so that the focus can be put either on the accumulation of a complete, coarse simulation model, or on the compilation of 'a priori' information for the design of identification experiments.

All the identified further approaches have a good potential of reducing the modelling effort significantly, with the latter two paths being particularly important, easy to realise and the first ones to pursue from the status quo of the current prototype implementation.

To round off this discussion on the modelling effort, it should be clearly pointed out that irrespective of any of the above considerations for further improvements, the developed combination *MODEL*^{ing}

+ fuzTF represents a very easy way to produce models of highly complex processes for any modelling situation. It is particularly advantageous in situations where there is no direct computer access to process data available. In fact, the approach is even the only viable way to nonlinear MIMO models for the envisaged user group - the area engineers - in the modelling situation defined at the beginning of this work. The need to address this particular modelling situation is a good example for the general importance of research into modelling on the basis of knowledge which is briefly addressed, among other issues, in the following section.

8.5 The Overall Results Matched Against Expectations

The importance and value of knowledge as a major resource and production factor has been discovered in many fields - not least in automation technology. Knowledge that is directly implemented in automation systems, however, is quite inflexible with respect to its application. Models as a means of building up, storing and flexibly applying knowledge about systems will therefore become increasingly important in the near future, as Jobling [128] predicts. The work presented in this thesis was aimed to be a step in this direction.

The overall expectation was a new, integrated approach to process modelling on the basis of knowledge and experience from area engineers which had to be designed in such a way that it can be easily understood and handled without any particular experience in process modelling.

As a whole, the expectations of the work, which were further detailed in Chapter 1, have been fulfilled. They have, in fact, been exceeded with respect to the efficient handling of highly nonlinear multivariable processes due to the successful 'fuzTF' approach. With the fuzzy hybrid modelling taking both more time to develop and a more predominant role in the project than could have been foreseen, however, the work had to be focused on heuristics based modelling as was discussed in section 8.3.

Bearing in mind that the implementation of the overall approach is only a prototype, all test results have also been very satisfactory. The experiences gained with the implementation were very valuable keys for deriving further improvements which have either already been implemented or are proposed as future extensions in Chapter 9.

The generally good match between the goals of the project and its results shows that the original expectations were justified and that further work in this direction is promising.

8.6 The Outcome Of This Work In The Context Of The Collaborative Project

In the context of the collaborative research project, the knowledge engineering approach to process modelling was meant to take the role of a pre-processor to the other modules, in particular to the module for experimental process identification, which is being developed. The developed *MODEL*^{ing} approach and its prototype implementation fulfil this prospective role by providing all relevant information via simulation code and the ASCII Report file to other CACSD modules. Particularly important parts of the "a priori" knowledge required for the design of identification experiments, which are therefore acquired through *MODEL*^{ing} and summarised in the interface files, are:

- information on the process inputs and outputs
- information on the nonlinear influences and their relative importance
- the definition of the state space to be investigated (i.e., the range of operating conditions for all process parameters)
- information about specific restrictions (e.g. limited rates of change, process specifications like tolerances)
- information on disturbances
- decisions on the applicability of specific types of test signals, taking process specifications into account
- the likely transfer function structure, possibly complemented with approximated parameters
- attributes, which give a detailed account of the static and dynamic properties, especially with respect to nonlinearities (e.g., 'multiple variable' or 'discontinuous')

Further to the preparation of the design of experiments, the developed approach facilitates the direct application of the coarse process models that are possibly derived in the case of sufficient information. An initial, coarse simulation is one of the particularly interesting direct applications. The advantage of this facility is the possibility to get a first feel for the effects of nonlinear process properties in the dynamic simulation, which could be the key to important decisions in the design of experiments. Nonlinear effects, for example, which hardly influence the dynamic process behaviour, can be neglected in the numerical process identification. With *MODEL*^{ing}, the real process may not even exist (e.g. in design phase), yet it could be modelled as long as experience from a similar process is available. Finally, a directly derived nonlinear model could be very useful in the optimisation and test of simple - especially linear - controllers. The possible coarseness of models derived through *MODEL*^{ing} is not a problem in the latter case since the optimisation would focus on finding the best 'compromise' parameters of the simple controller to cope with different process conditions rather than aiming at a parametrically correct nonlinear controller.

9. Conclusions And Further Work

Highlighting the thread running through it, the thesis is firstly summarised in this final Chapter. Afterwards, the conclusions of the work are drawn and the possible further extensions of this project are detailed.

9.1 Summary Of The Thesis

After establishing the need for simplified modelling approaches that make use of the knowledge, which area engineers accumulate in the course of their everyday work with industrial processes (Chapter 1), a brief review of the main types of modelling support was given in Chapter 2. At the end of Chapter 2, the importance of developing the knowledge acquisition procedure was stressed along with the need to consider modelling approaches on the basis of partial knowledge. This latter field was reviewed from a control engineering point of view in Chapter 3. Despite its deficiency with respect to representing dynamic system properties, fuzzy modelling was found to be the most appropriate 'qualitative' approach for the control engineering domain. In Chapter 4, this deficiency was addressed with the introduction of a new fuzzy hybrid model, which is geared at modelling complex nonlinear, multivariable *dynamic* processes. With Chapter 5, the work returned to the 'overall' perspective of developing a knowledge acquisition approach to process modelling, which was indicated in Chapter 1 and specified in Chapter 2. The focus of Chapter 5 was therefore the structured knowledge elicitation procedure which considered in particular the newly introduced fuzzy hybrid modelling approach as an integral part. OMT-diagrams were used for the design and graphical illustration of the procedure and all major design considerations were summarised. An additional part of Chapter 5 described the design of an appropriate, flexible knowledge representation. In Chapter 6, the knowledge acquisition procedure was brought to life by illustrating its prototype implementation. Apart from being a test-bed for the development of a self-explanatory GUI, this prototype served the purpose of validating the procedure that was proposed in Chapter 5. With the implementation of a MATLAB code and ASCII-Report generator, a prototype path for the integration of the process knowledge, which is built up through the developed approach, into other CACSD modules was developed, too. Chapter 7 summarised some experiences with the application of the prototype implementations and is therefore likewise a part of the validation work. In Chapter 8, finally, the complete work was discussed with respect to the expectations and results, the earlier design decisions in retrospect and the research methodology employed.

9.2 Conclusions

This work is a step towards modelling on the basis of the industrial process expert's experience in order to facilitate the systematic application of their knowledge for control engineering purposes. For companies where control engineering experts are available, the work lays the ground for a new means of efficient knowledge-communication between process engineers and control engineers. Further to these aspects of knowledge based modelling in industry, its importance is emphasised by the fact that real processes can often neither be modelled analytically nor identified experimentally with respect to all their nonlinear properties. Therefore, they are normally approximated with models of relatively low complexity. Although such approximations are normally sufficient, it is often helpful to be aware of the variety of possible effects of further nonlinear influences. The knowledge about these influences is frequently part of the area expert's experience from working with the processes. With the introduced knowledge engineering approach, this information can now be tapped.

The knowledge based modelling approach as it is presented in this work can handle

- time continuous,
- nonlinear,
- multivariable,
- dynamic systems with
- distributed parameters.

Linear, singlevariable and static systems as well as those with concentrated parameters are handled as sub-sets of this scope. Discrete systems, however, cannot be handled and should therefore be addressed in the further work.

The main novelties that resulted from this contribution are:

1. The fuzzy-hybrid modelling approach for nonlinear multivariable *dynamic* processes, which builds on partial information about the global system behaviour in the form of locally valid linear single input / single output - approximations. This approach complements the successful fuzzy hybrid approach by Takagi-Sugeno which builds on the same kind of modelling information but is limited to handling static relationships. The unique advantages of the suggested fuzzy-hybrid modelling approach compared with other existing modelling techniques that are based on partial process information have been elaborated in detail in Chapter 4.

2. A new modelling approach that is based on descriptive attributes referring to the process behaviour. This approach facilitates the use of particularly abstract and not quantifiable information. It will form an important part of the 'a priori' knowledge in the design of process identification experiments.
3. The systematic and structured knowledge acquisition approach to process modelling, which addresses the overall aim of this research project. According to this aim, the sequential succession of modelling steps was carefully designed to bring the uninitiated user increasingly into the context of process modelling. The expected information per step in the knowledge acquisition procedure can either be used directly or after a translation into control engineering terminology for the build-up and storage of the user's experience. This knowledge can be passed on and integrated into the other CACSD modules currently being developed within the collaborative project. Fuzzy hybrid modelling according to the first major contribution (1. above) and attributes-based modelling (2. above) are important integral parts of the knowledge acquisition approach.

9.3 Further Work

This final section of the thesis points the way to possible extensions of this work. The variety of paths to pursue confirms how wide open this field is for further work. In fact, more and more interesting directions appeared to open up during the work on this project; the further work suggested below is therefore only a summary of the extensions particularly closely related to the work presented in this thesis.

Overall, the goals for further work on the knowledge engineering approach to process modelling must be

- the extension of its scope (in particular to address all kinds of knowledge),
- the improvement of the applicability (further refinement of the self-explanatory handling), and
- the realisation of its envisaged role as an "open tool", which provides the acquired information not only to the other modules within the collaborative project, but also to various external - academic and commercial - CACSD packages (open CACSD).

Additionally, some further investigations with respect to the novel fuzzy hybrid modelling approach and its properties should be carried out.

The specific ideas of possible extensions to this work are as follows:

9.3.1 Further Work On The Novel Fuzzy Hybrid Modelling Approach

A) Investigations On Possible Deductions From Fuzzy Static Characteristics

It appears to be very promising to investigate how far deductions with respect to the structure of conventional process models are possible on the basis of certain arrangements of the fuzzy hybrid model. Since the basic structure 'fuzSC-fuzTF' should generally be applied, 'arrangements' refers here mainly to the possible sources of influence parameters.

In addition to the deduction of a conventional 'Wiener' or 'Hammerstein' structure[1] in the case that the overall output or input, respectively, was found to be the only influence parameter that is fed into the fuzzy adapter, deductions from some multiple input / multiple output (MIMO) arrangements appear to be possible. Such MIMO processes can only be modelled by a combination of fuzzy hybrid models, since every fuzzy hybrid model can handle several inputs but only one output variable. In order to allow for such a translation into conventional MIMO structures, only either the main input or the output of a fuzzy hybrid model should be used as the single influence fed into another fuzzy hybrid model (example:).

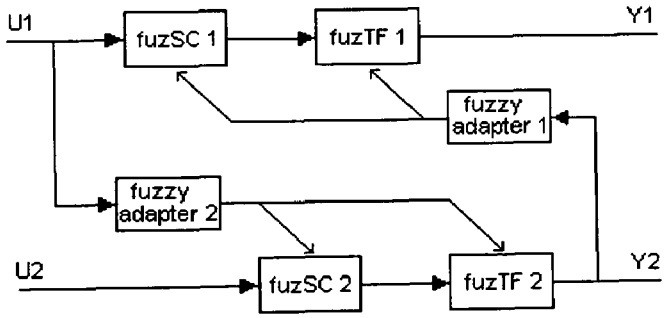


Figure 9-1: Example Of A MIMO Fuzzy Hybrid Model With One Influence Per Fuzzy Adapter

The idea behind such deductions is as follows. The fuzzy hybrid model appears to be closer to the understanding of the 'practical' modeller who knows how the process works. For example the influence of the water level in the funnel tank (Chapter 4) on its static and dynamic behaviour is quite obvious and the fuzzy influence parameter (level 'h') is therefore easily determined. The 'Wiener' type model structure, on the other hand, cannot be determined on the basis of practitioner's experience. However, such a structure could be determined from the fuzzy hybrid model and could be useful to enable a simplified linear controller design using an inverted static

[1] Isermann [51] gives an overview of conventional nonlinear structures.

characteristic for linearisation. Such simplified controller design approaches should be supported in a practical CACSD environment.

It is important to bear in mind, however, that neither the 'Wiener' structured nor any other conventional nonlinear model can represent, for example, the nonlinear dynamic aspects (like varying time constants) of the fuzzy hybrid model. Any kind of 'deduction' approach as suggested above is therefore normally only a simplification for controller design purposes. Prior to the online implementation, such simplified compensators and controllers must therefore be carefully tested against the complete nonlinear process model.

B) Inversion Of The Fuzzy Hybrid Model

Another interesting area of further work is the investigation of possibilities to invert the fuzzy static characteristic (fuzSC) or possibly even the whole fuzzy hybrid model, including the 'fuzTF' block. Used as a filter, for example, an inverted fuzzy static characteristic could be an invaluable tool to support nonlinear process identification experiments. Furthermore, the complete inverted model could be directly applicable as a nonlinear fuzzy hybrid controller.

An alternative to actually inverting the fuzzy hybrid model for control purposes could be the optimisation of PID controllers for each of the locally valid linear transfer functions of the process model. On the basis of the different PID parameters for the varying operating conditions, a fuzzy hybrid controller can easily be combined in the same way as the fuzzy hybrid model, with the adapter for the controller parameters using the same influence-parameters as the one in the model. The controller structure resulting from this model-based approach would be more comprehensive, considering possibly a variety of influences, than that of 'standard' fuzzy adapted PID controllers.

C) Studies On The Stability Properties Of The Fuzzy Hybrid Model

The stability of fuzzy controllers is hard to prove and despite some recent contributions still an active field of research. In the case of the fuzzy hybrid model, however, the stability analysis situation is related more closely to that of adaptive linear controllers. Such a study would obviously be very interesting and important, although the experiences with respect to stability have been positive. Judging from the test results, the approach has very stable parameter interpolation properties (cf. Appendix A4). Furthermore, the stability of the locally valid linear transfer functions, which form the basis of the fuzzy hybrid model, can easily be checked using conventional techniques. The stability of the linear models with intermediate parameter settings could be checked in a similar fashion, which could be automated. Beyond this separate consideration of the components that make up the overall fuzzy hybrid model, the stability study would have to focus

therefore on the complete system, including the interaction of the fuzzy adapter with the transfer function.

D) Automated Validation Of The Estimated Relative Importance Of The Influences
After completing the fuzzy hybrid model, it is possible to derive automatically information about the correctness of the user's initial judgement with respect to the influences ranking and the selection of the MAIN input. This information should be obtained by analysing the parameter variations along the columns and rows of the rule-base matrices to find out which influence is stronger.

9.3.2 Extensions To Address All Types Of Knowledge

As was mentioned before, the presented approach to knowledge based modelling, which focused on heuristic knowledge, should be extended to address all relevant types of knowledge and experience. The appropriate integration of causal, case-based and probabilistic knowledge into the modelling approach was already outlined in Chapter 5.

The ***causal*** knowledge acquisition sequence would support systematically the specification of mathematical equations that theoretically describe the relationships between the relevant variables of the modelled component. One of the possible support mechanisms that facilitate the derivation of equations which are not directly known to the user is the library-based approach, in which pre-programmed elementary sub-components (with underlying equations) can easily be accumulated to the overall component to be modelled.

Since causal knowledge forms the traditional basis of modelling approaches, its exploitation within the knowledge acquisition approach would in particular open up possible links to many existing tools for further evaluation. Links to the object-oriented modelling languages Omola and Dymola would be of particular interest in this respect. Once the model equations are transferred to Dymola, for example, various manipulations are possible and code for the simulation in various environments - like ACSL and Simnon - can be generated. Since Dymola reads and writes also models in the neutral DSblock-format, the latter appears to be the most suitable path of communicating causal knowledge acquired through the extended *MODEL*^{ing} approach.

A particularly interesting field of work with respect to extending the knowledge acquisition approach to causal modelling would be the simplified evaluation of incomplete or partial knowledge. Essentially, this requires the link-up of two so far distinct strands of research. Firstly, intelligent causal modelling and secondly, qualitative modelling. Such work could overcome the great

difficulties in the preparation of qualitative causal models, which have been described in Chapter 3. Mycroft or QSIM (with its more recent extensions) are the most promising candidates in this respect as was also discussed in Chapter 3.

With respect to the utilisation of ***probabilistic*** knowledge, the 'ROSA' approach, developed by Krabs [409], could be employed. 'ROSA' was designed for the generation and validation of heuristic models on the basis of measured data and a priori knowledge. Within the context of the integrated knowledge engineering approach to process modelling, however, existing process data from quality assurance records - like Statistical Process Control (SPC) - rather than specifically acquired data should be applied. This type of data has not yet been systematically evaluated for process modelling.

The use of ***case-based*** knowledge is not explicitly featured in a separate module within the *MODEL*^{ing} concept (Chapter 5). Instead, this knowledge type should be exploited via a library-based approach that runs as a supportive tool concurrently to the 'causal' and 'heuristic' modules, providing on request information in the form of typical answers given during previous modelling sessions for similar process types. This supportive library would therefore allow for analogous reasoning within the approach and hence simplify the modelling of different versions of a process component.

9.3.3 Extensions To The Knowledge Acquisition Approach And Its Implementation

Firstly, of course, the knowledge acquisition approach ought to be implemented to the full extent of its design, which was introduced in Chapter 5 (in particular all nonlinear static modelling remains to be implemented).

In addition to the simplified identification algorithms that have been implemented in the prototype (e.g. tangent approach, according to Strejc [51]), a variety of likewise easy-to-use approaches should be offered as alternatives. Algorithms that are based on characteristic rise times - the times it takes for the output to rise to certain percentages of its steady state after a step input -, for example, are suitable alternatives [129]. Also, it would be sensible to offer identification approaches based on ramp or impulse responses in order to increase the flexibility and user-orientation of the program. Obviously, this extension is mainly an implementational question rather than a matter of research.

Important directions of research to pursue from the current status are the promising extensions to improve the self-explanatory handling of the implemented approach for uninitiated users, which were discussed in detail in Chapter 8.

To improve the results in situations where the user is able to specify nonlinear characteristic 'effects' (i.e., particular properties/ shapes; maybe graphically) in the nonlinear static (NLs) modelling sequence but cannot specify the associated data, it would be useful to show the collection of characteristic effects at the end of the NLs section graphically to the user, who would then be requested to select the figures in the sequence they appear with increasing input value. Using graphics manipulation techniques, the figures could then be 'snapped' together to a single qualitative static characteristic. Although without reference to numerical values, such coarse, qualitative shapes of static characteristics could be a valuable help in further analysis and design steps.

The modelling of discrete event systems should be included in a simple extension to the (fuzzy) rule based modelling approach within the nonlinear static modelling procedure.

Different to the current prototype, the final version of the *MODEL*^{ing} implementation will have to address the issue "open CACSD" very carefully. The neutral model bus "DSblock" [14, 15], which was mentioned before, is certainly one of the particularly important data exchange formats to be considered - at least as far as mathematical system descriptions are concerned. Furthermore, the suggestions by Varsamidis et al. [130] and Barker et al. [131] ought to be considered and the currently still ongoing work on an "Object-Oriented Information Model For Intelligent Modelling" by Li et al. [33] could also give some interesting inputs to the extension of this project. Any future links to other CACSD modules should be implemented as bi-directional links, so that components that are altered in another module can be read back into the knowledge-based modelling tool.

Appendix To Chapter 1

The Needs In Industry - A Summary

Both a literature review of current CACSD developments and a direct 'hands-on' overview of the existing CACSD programs at the VDI-workshop[1] gave a fairly clear picture of the deficiencies with respect to the support of industrial users - especially for those users who are not regularly working in this field. To complement these impressions, contacts to industry were made and questionnaires were sent out. The idea was not to obtain a statistical basis for an exhaustive survey but rather to carry out some spot checks to verify the identified needs. A comparatively small number of questionnaires (26) was therefore sent out to users of control systems from different branches of manufacturing industry. Also, two manufacturers of control gear and control engineering departments in chemical and steel industry were visited to discuss the collaborative overall project of a simplified computer aided control system design program of which the modelling approach of this work forms a part.

The main results of the discussions and the evaluation of replies to the questionnaire are summarised in the following:

- basic Single Input / Single Output (SISO) control systems are structurally designed together with the process design but not theoretically determined and analysed (manual adjustment during process operation)
- advanced control systems are normally designed after the new process has been in operation because of the lack of process models
- mathematical process models are usually applied - but only in special control engineering departments and for some of the process components; static models for complete plants are also sometimes applied while dynamic models for more complex plants are virtually non-existent
- in control engineering departments, theoretical and experimental modelling techniques (i.e. identification) are applied
- the experience of area engineers is not used for modelling
- modelling tools which make use of the area engineer's knowledge by translating it into appropriate models and thus overcoming the information-gap between practitioners and the control engineering department are still missing
- the main shortcoming of computer aided control system design (CACSD) programs is seen in the lack of modelling support - especially the pre-selection of model structures on the basis of

[1] VDI-Workshop "Regelungstechnische Programmpakete" ("Control Engineering Programs"); March 1993 in Düsseldorf; Chair: Prof. Dr.-Ing. R. Schumann

limited a priori knowledge and the appropriate experiment design are major difficulties; even specialists loose the overview of suitable techniques in the case of more complex processes (e.g. non-linear, multi-variable)

- a likewise important problem with respect to CACSD programs is the exchange of data and results between different programs (non-standard interfaces)
- the need for qualitative modelling and simulation techniques is felt for the purpose of a first, general analysis of the process behaviour
- the interest in model libraries is mixed: some industrial users would appreciate a library with detailed models of their applications with the possibility to adjust parameters using process data and others reject the idea of a library approach because of the effort to adapt pre-defined standard models to their application
- processes are generally treated as SISO; multi-variable techniques are still rarely employed although there is a clearly increasing trend
- although virtually every process in industry exhibits non-linearities of some kind, these problems are usually dealt with by linear approximations; adaptive control is very rarely applied
- artificial intelligence and fuzzy control approaches are very rarely applied - often only in research departments
- in companies without a special control engineering department, design and tuning of control systems is normally not done systematically; simulation is not undertaken

Appendix To Chapter 2

Knowledge Types And Modelling

In Artificial Intelligence, the following types of knowledge are distinguished [45]:

- *Heuristic knowledge* is knowledge which points from problem features to problem solutions. Heuristic knowledge which can be formulated in rules of different 'granularity' - from global relations to local if-then rules - is easy to interpret, but rarely applicable across different domains.
- *Model-based (causal) knowledge* is knowledge about general relationships between problem solutions and problem features that can be used in different ways like fault diagnosis or simulation. It is, however, more difficult to formulate the appropriate relationships.
- *Statistical (probabilistic) knowledge* is uncertain knowledge which is gained from statistical analysis of a set of successfully solved cases.
- *Case-based knowledge* is knowledge about successfully solved cases: a set of problem features with the correct problem solution, which is used for analogous reasoning. A solution to a new problem is here directly adopted or modified from a known case which comes as close as possible.

Heuristic knowledge is sometimes referred to as '**shallow**' knowledge because it does not normally represent the causal nature of an observation and because predictions can only be made for events that were considered in the formulation of the knowledge base. '**Deep**' knowledge, referring to *causal* knowledge, can be used to provide both explanations and predictions in new situations. It must not be overlooked, however, that the *causal* knowledge representation in turn has some disadvantages compared with *heuristics*:

- All governing relations must be known mathematically and programmed after some sort of manipulation (i.e., abstraction) for the device under consideration.
- Experience gained with the real process cannot be expressed in a 'deep' knowledge model because *causal* knowledge bases are inflexible with respect to the integration of different kinds of knowledge.

Traditionally, modelling focuses mainly on *causal* knowledge and employs some *case-based* knowledge in the process of building the *causal* model. This is because *causal* knowledge is directly applicable as a model, because it can be easily handled and because it is very suitable for numeric simulations. Ideally, however, all types of knowledge should be exploited for process modelling - especially in situations when only little or no *causal* knowledge is available.

Appendix To Chapter 4

Test Of The Interpolation Properties Of The Fuzzy Static Characteristic

In the following, the interpolation qualities of the fuzzy static characteristic as part of the fuzzy hybrid model are investigated. The default simplifications have been applied. For this test, four benchmark problems have been defined as follows:

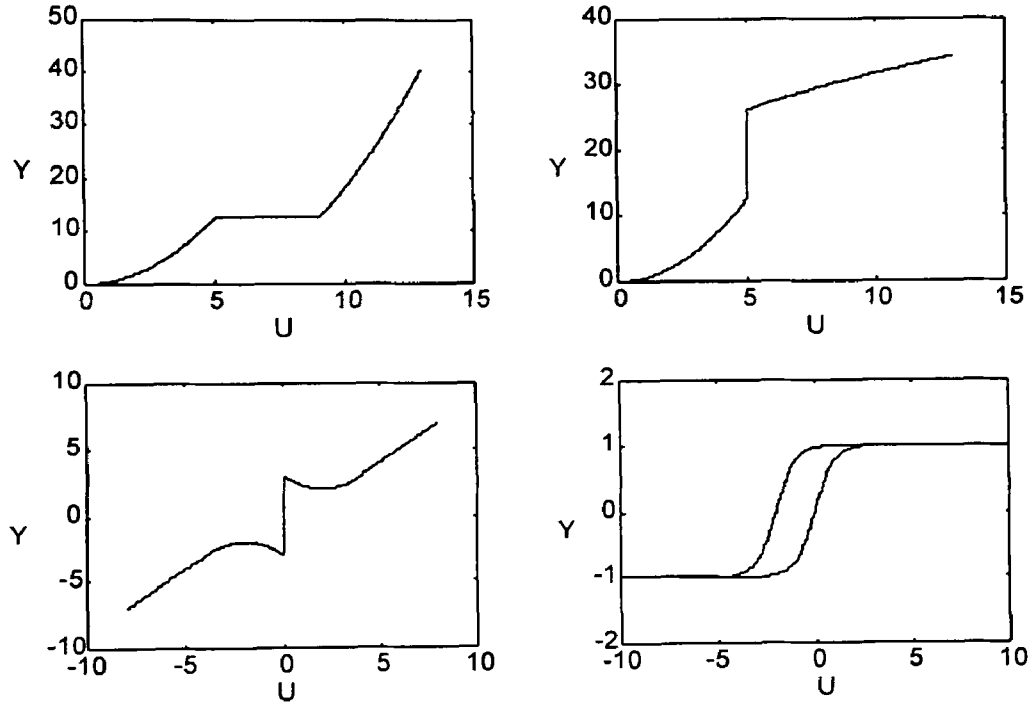


Figure A4-1: Four Static Characteristics As Benchmark Problems

The four benchmark problems cover most awkward conditions (e.g. discontinuity and multiple valued) with respect to interpolation that are found in typical static characteristics.

Since most interpolation approaches can give quite good results in the case that many data points are provided, the main concern was to define only a minimum amount of points for the interpolation task - an arc, for example, is defined by three points. The pre-defined points are marked in the following interpolation results with circles while the interpolations as such are plotted as dotted lines to distinguish them from the ideal shape of the static characteristics (full line).

In order to be able to judge its interpolation quality, the 'fuzSC' is compared with linear and cubic spline interpolation results.

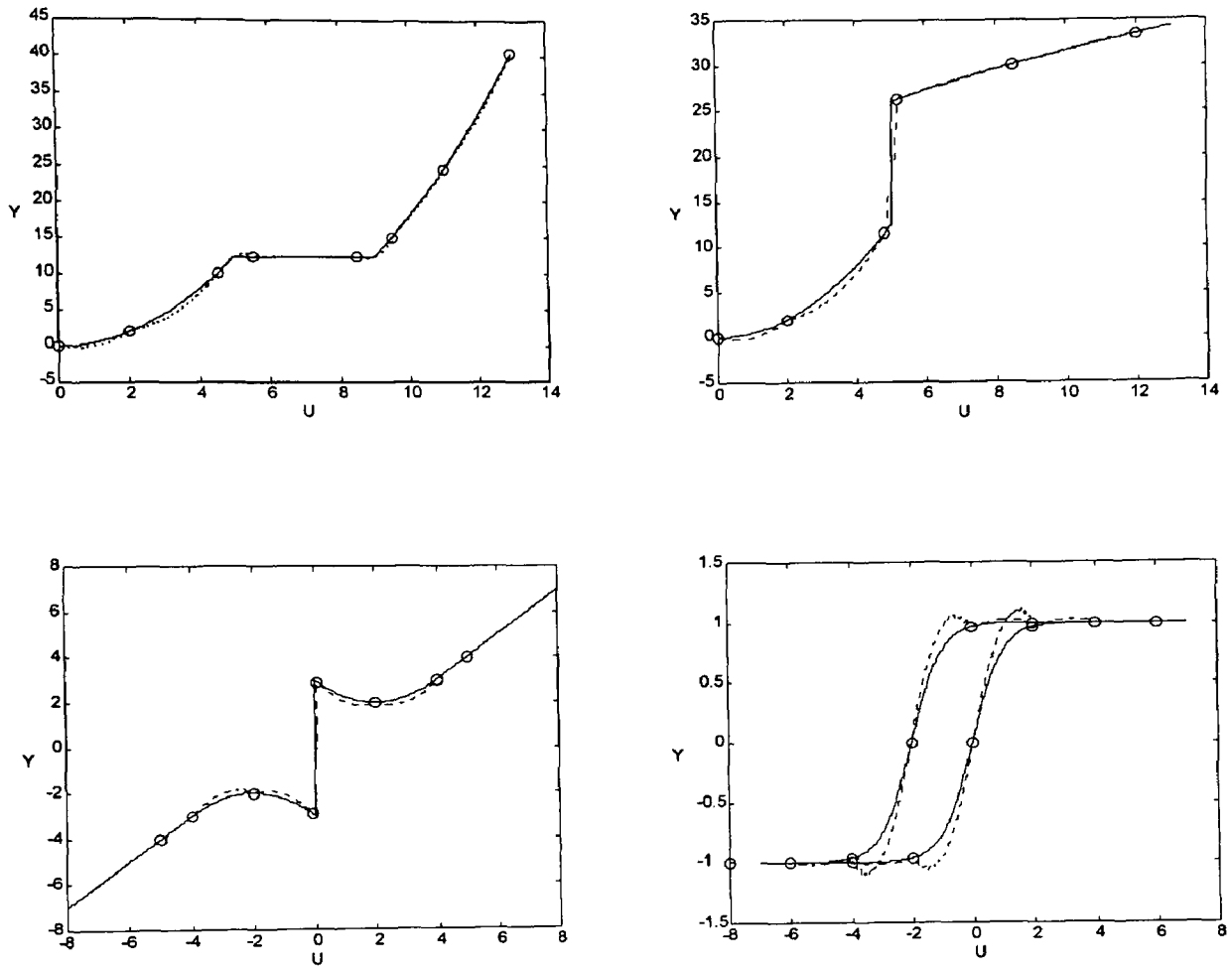


Figure A4-2: fuzSC Interpolation Results

Overall, the fuzSC interpolation results are very satisfactory and, in fact, most convincing among the three interpolation approaches:

Benchmark problem 1 did not pose much of a problem to any of the interpolation algorithms and was solved slightly better by fuzSC than by the two reference approaches. The problems 2, 3 and 4 proved to be beyond the scope of the cubic spline interpolation. In problem 2 and 3, the fuzzy static characteristics are of similar quality as the linearly interpolated ones.

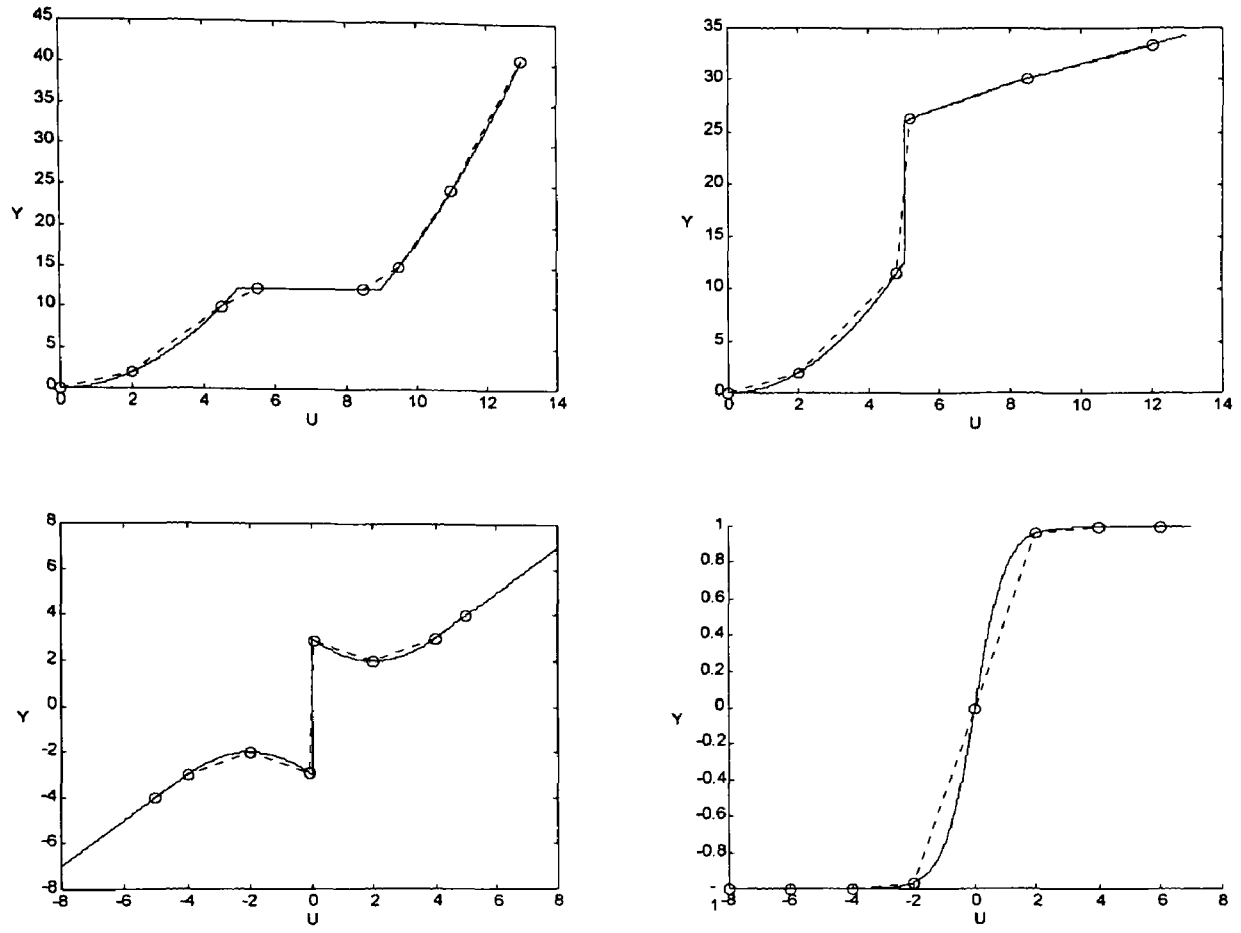


Figure A4-3: Linear Interpolation

Using the linear interpolation, a single branch of the hysteresis in the shape of the tanh(x) function (problem 4) can be approximated in almost the same quality as with the fuzSC approach. The interpolation errors for both approaches could be significantly reduced by supplying only two more data points per hysteresis branch. In order to model the complete hysteresis on the basis of linear interpolation, however, the combination with boolean logic, a toggle switch between two separate linear interpolators or any other additional construct would be required. Here lies one of the particular strengths of the fuzSC approach, which is flexible enough to handle additional influences (here direction of change) within the same approach. The consideration of multiple valued characteristics in a single interpolation approach greatly simplifies a standard modelling approach, which is of particular importance for less experienced modellers.

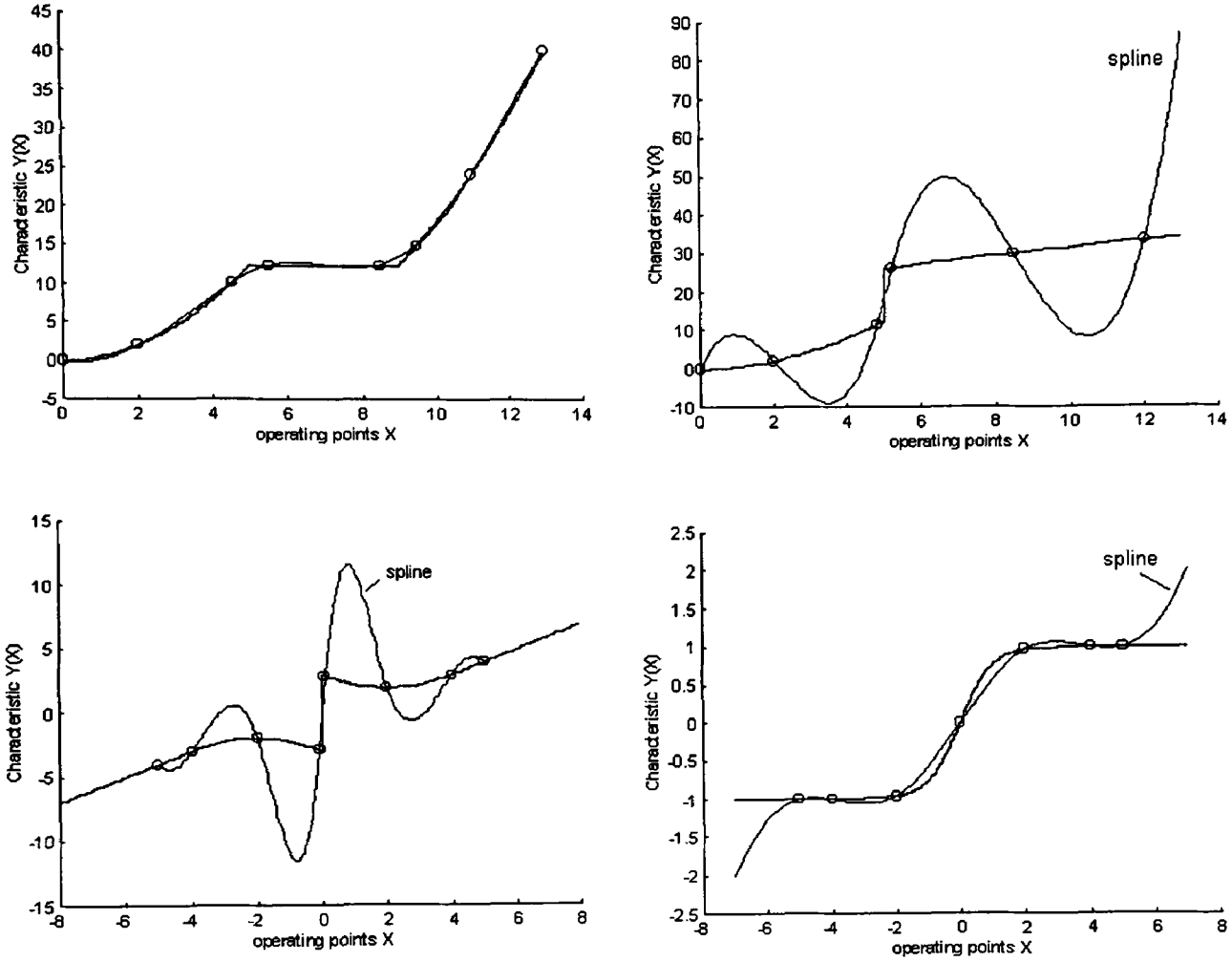


Figure A4-4: Cubic Spline Interpolation

Appendix To Chapter 5

Data To Be Acquired Using The *MODEL*^{ing} Approach - Sorted According To The Object Model As Attributes Of The Key Classes

Component:

(all 'central' information on the Component and its overall Model)

- name of component
- process domain (e.g. Chem., Elec., Mech., ...)
- process type (e.g. pump, gears,...)
- known sub-components (names)

model documentation according to Simulation Council [118, 123]:

* version-no. of model
* statement of purpose of the model
* modelling assumptions
* range of conditions for which the comp. model has been tested
* range of conditions for which agreement between model & reality has been obtained
* description of validation tests
* comments

Terminals:

- number (implicitly via number of instances)

(attributes that Input, Output and Disturbance have in common:)

- name
- symbol
- units
- operating points (several per variable, if applicable)
- operating range (min./max.)
- critical conditions:
  - max. rate of change
  - alarms for any variable
  - tolerances for process parameters during normal operation
    (according to production rules or standards)

- (medium - only for chem. processes!)
- transducers (types, e.g. PT100 → to be linked to library of typical characteristics)
- i/o signals measurable / directly accessible?

- **additional Input attributes:**
  - limitations of actuators (ranges, rates of change)
  - staircase signal possible?...step?...impulse? (= preference sequence according to usefulness)
  - sequence of impulses possible? (→PRBS)

- **additional Output attributes:**
  - MAIN input with respect to individual output
  - 'ranking' of influences according to their importance

- **additional Disturbance attributes:**
  - disturbance acting on input, output or directly on process?
  - possibility to apply test-disturbances?
  - directly measurable?

MODEL:

Heuristic Model:

MIMO, MISO, SISO

MIMO is composed of MISO which in turn is split into SISO components. Most instances in the Data model refer to SISO information - the overall component model (MISO or even MIMO) results from associations between the SISO parts. Systems with distributed parameters are also modelled as MISO.

NLdAttrib: (nonlinear dynamics attributes)

- nonlinear dynamic behaviour? (Y/N)
- limited rate of change: - min. and/or max. limit
  - invariant / variant
- dead time: - invariant / variant
- variant time constant
- variant damping ratio ....and all negations to these attributes

LinAttrib: (linear dynamics attributes, related to step response)

- steady state
- oscillates or overshoots
- oscillates (complex poles)
- responds without lag
- increasing speed of response
- decreasing speed of response
- non-minimal phase

....and all negations to these attributes

LinMath:

Parametric transfer function structures and associated parameters (physically meaningful: gains, time constants, damping constants, etc. / or polynomial parameters: ai, bi)

P = Proportional action
I = Integral action
D = Derivative action
T = Time constant; Tn = lag of order n (e.g. T2 = 2^{nd} order)

- **Proportional**:	PT1	Kp, T	/	a0, a1, b0
	PT2	Kp, T, D	/	a0, a1, a2, b0
	PTn	Kp, TN	/	a0...aN, b0
- **Integral**:	I	Ki	/	a0=0, a1, b0
	IT1	Ki, T	/	a0=0, a1, a2, b0
	PIT1	Kp, Ki, T	/	a0=0, a1, a2, b0, b1
	PIT2	Kp, Ki, T, D	/	a0=0, a1, a2, a3, b0, b1
	PIDT1	Kp, T, Tn, Tv	/	a0=0, a1, a2, b0, b1, b2
- **Derivative**:	PDT1	Kp, Tv, T	/	a0, a1, b0, b1
	PDT2	Kp, Tv, T, D	/	a0, a1, a2, b0, b1

Additionally, all characteristic step response parameters, which are required for the simplified parameter identification, must be handled by purpose-built slots.

ProdRuleSet:

- set of production rules in multiple value slots: *IF* <condition> *THEN* <consequence>
- data, defining the linguistic values (either 'crisp' or 'fuzzy' definition - 'crisp' to be handled as special case of fuzzy)

NLSC:

- look-up table (static relation between absolute data of input, U, and output, Y)

NLsAttrib:

(nonlinear statics attributes; possibly supplemented with data, which relates the attributes to specific operating conditions; additionally, any available data is transferred to NLSC, where it is summarised to a look-up table)

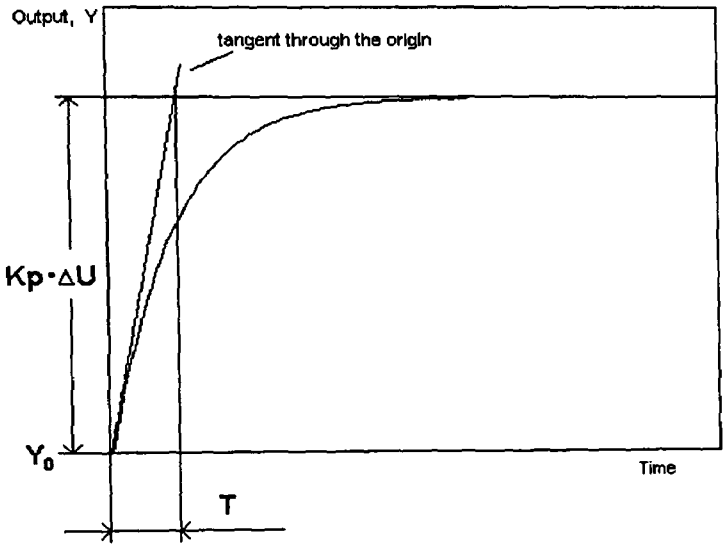
- static nonlinearity?
- conventional nonlinear representation as Wiener-structure? / Hammerstein-structure? (possibly deduced from fuzTF representation - cf. further work, Chapter 9)
- single/multiple valued
    - multi-valued (depending on input change sign(du/dt))
    - multi-variable (additional influence)
    - data
- time invariant / time variant
    - time since process start / daytime / calendar time / other..
    - relevant time scale
    - data
- (dis-)continuous
    - number
    - kind(s)
    - where? (data)
- positive and/or negative slope of SC
    - partly linear (→ data)
    - data (min. 3 points)
- saturation, upper/lower bound
    -data: input-range, associated output, adjacent SC-data
- absolute or relative MAX / MIN
    - number
    - data pairs

- intermediate dead zones
    - number of such dead zones
    - data: range of input/output

Appendix To Chapter 6

The Implemented Simplified Identification Equations And Lookup Tables That Are Based On Responses To The Step Input ΔU

PT1:



$$G(s) = \frac{Kp}{1 + T \cdot s}$$

$$Kp = \frac{Y_\infty - Y_0}{\Delta U} = \frac{(Kp \cdot \Delta U)}{\Delta U}$$

Figure A6-1: "PT1" Identification

PTn:

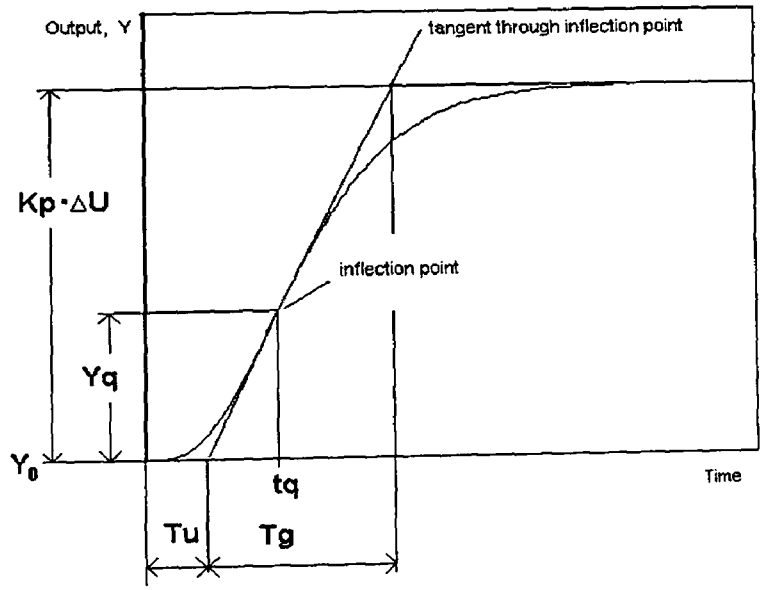


Figure A6-2: "PTn" Identification

$$G(s) = \frac{Kp}{(1 + T1 \cdot s)(1 + T2 \cdot s)...(1 + TN \cdot s)}$$

$$Kp = \frac{Y_\infty - Y_0}{\Delta U} = \frac{(Kp \cdot \Delta U)}{\Delta U}$$

For 0.104 ≤ Tu/Tg < 0.85:

Approximation on the basis of proportional transfer functions of 2nd to 10th order with standard time constant, T1≈T2=...=TN, and damping D = 1, according to Strejc [51]:

Enter the following lookup table in column 2 and determine the row with the closest match for the given Tu/Tg ratio. Order N = first column; determine T from 3rd, 4th and 5th column with the given values tq, Tg and Tu. TN is the average of these three T values.

Order, N	Tu/Tg	tq/T	Tg/T	Tu/T	Yq/(Kp*U)
1	0	0	1	0	0
2	0,104	1	2,718	0,282	0,264
3	0,218	2	3,695	0,805	0,323
4	0,319	3	4,463	1,425	0,353
5	0,41	4	5,119	2,1	0,371
6	0,493	5	5,699	2,811	0,384
7	0,57	6	6,226	3,549	0,394
8	0,642	7	6,711	4,307	0,401
9	0,709	8	7,164	5,081	0,407
10	0,773	9	7,59	5,869	0,413

Table A6-1: "PTn" Identification For 0.104 ≤ Tu/Tg < 0.85 According to Strejc

For 0 ≤ Tu/Tg < 0.104:

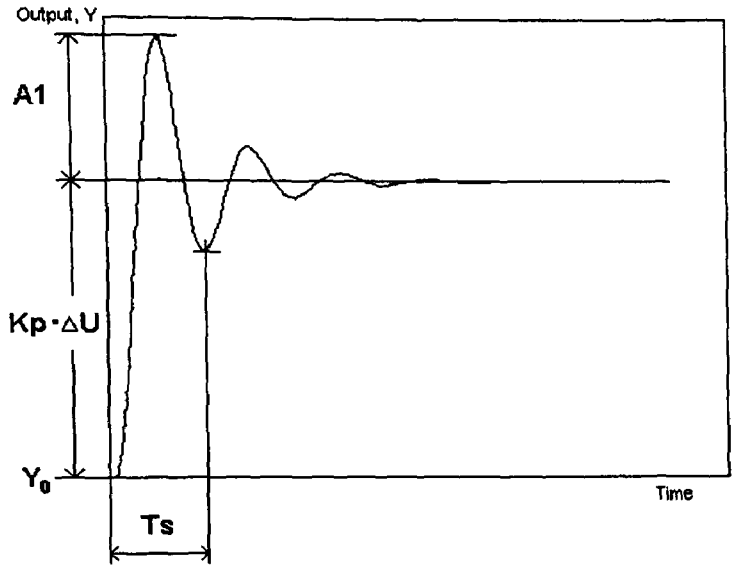
Approximation on the basis of a proportional transfer function of 2nd order with different time constants, T1 and T2, and damping D > 1, according to Strejc [51]:

Enter the lookup table in column 2 and determine the row with the closest match for the given Tu/Tg ratio. Determine T1 from 3rd column with the given value Tg; afterwards T2 from 1st column.

T2/T1	Tu/Tg	Tg/T1
0	0	1
0,1	0,05	1,3
0,2	0,07	1,5
0,3	0,08	1,7
0,4	0,09	1,85
0,5	0,0995	2
0,6	0,1005	2,15
0,7	0,101	2,3
0,8	0,103	2,45
1	0,104	2,75

Table A6-2: "PTn" Identification For 0 ≤ Tu/Tg < 0.104 According to Strejc

PT2:
(with damping D < 1)



$$G(s) = \frac{Kp}{1 + 2DT \cdot s + T^2 \cdot s^2}$$

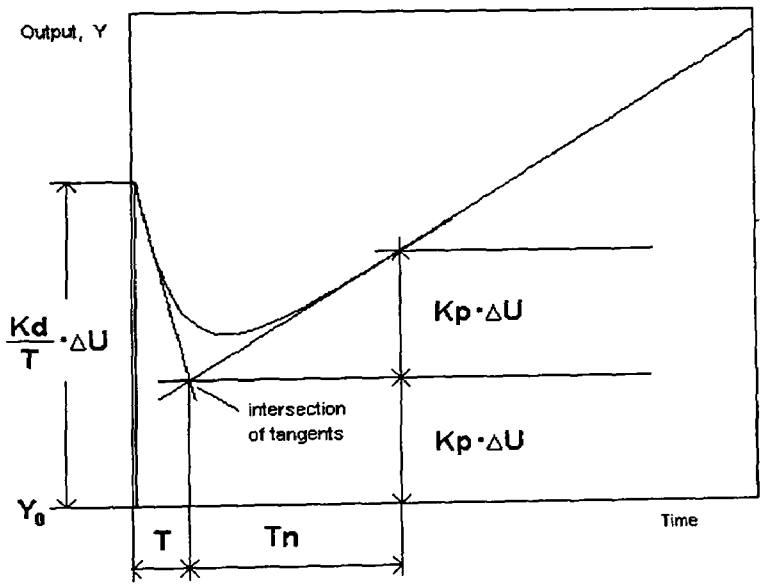
$$Kp = \frac{Y_\infty - Y_0}{\Delta U} = \frac{(Kp \cdot \Delta U)}{\Delta U}$$

$$D = -\frac{\ln \frac{A1}{Kp \cdot \Delta U}}{\sqrt{\pi^2 + \left(\ln \frac{A1}{Kp \cdot \Delta U}\right)^2}}$$

$$T = \frac{Ts}{2 \cdot \pi}\sqrt{1 - D^2}$$

Figure A6-3: "PT2" Identification

PIDT1:



$$G(s) = \frac{Kp \cdot \left(1 + \frac{1}{Tn \cdot s} + Tv \cdot s\right)}{1 + T \cdot s}$$

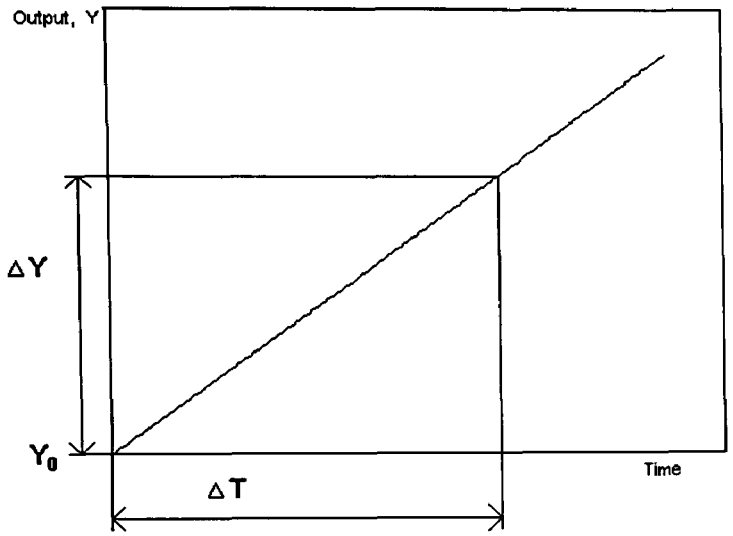
$$Kp = \frac{(Kp \cdot \Delta U)}{\Delta U}$$

$$Kd = \left(\frac{Kd}{T} \cdot \Delta U\right) \cdot \frac{T}{\Delta U}$$

$$Tv = \frac{Kd}{Kp}$$

Figure A6-4: "PIDT1" Identification

I:

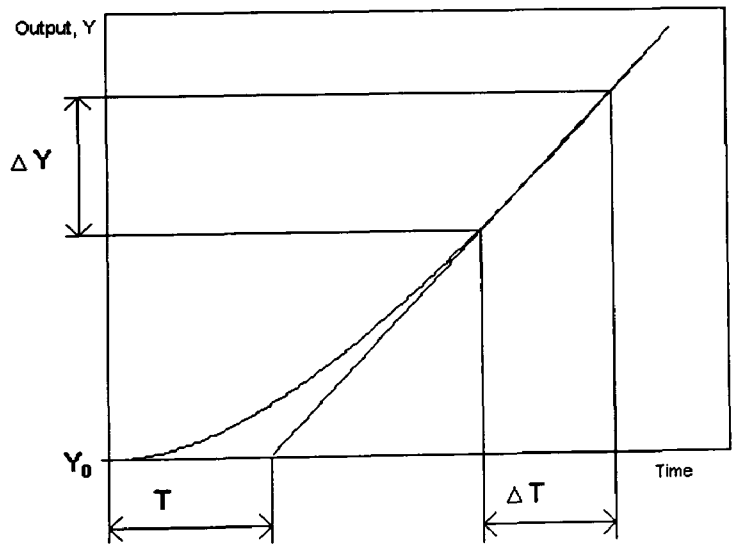


$$G(s) = \frac{Ki}{s}$$

$$Ki = \frac{dY}{dT \cdot \Delta U}$$

Figure A6-5: "I" Identification

IT1:

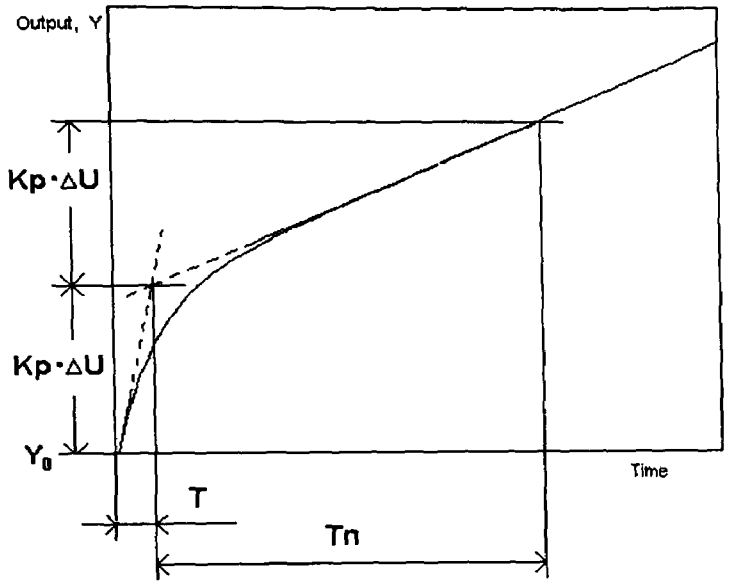


$$G(s) = \frac{Ki}{s \cdot (1 + T \cdot s)}$$

$$Ki = \frac{dY}{dT \cdot \Delta U}$$

Figure A6-6: "IT1" Identification

PIT1:



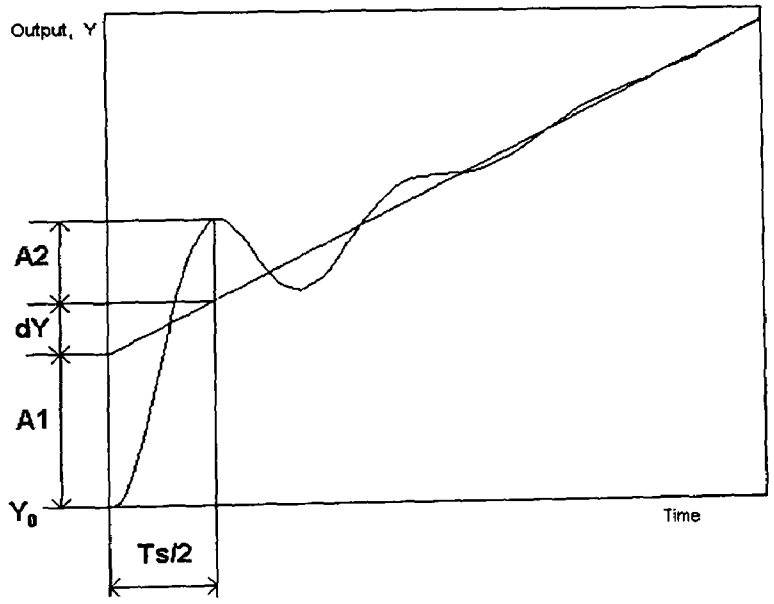
$$G(s) = \frac{Kp + \frac{Ki}{s}}{1 + T \cdot s}$$

$$Kp = \frac{(Kp \cdot \Delta U)}{\Delta U}$$

$$Ki = \frac{Kp}{Tn}$$

Figure A6-7: "PIT1" Identification

PIT2:



$$G(s) = \frac{Kp + \frac{Ki}{s}}{1 + 2DT \cdot s + T^2 \cdot s^2}$$

$$Ki = \frac{dY}{(Ts/2) \cdot \Delta U}$$

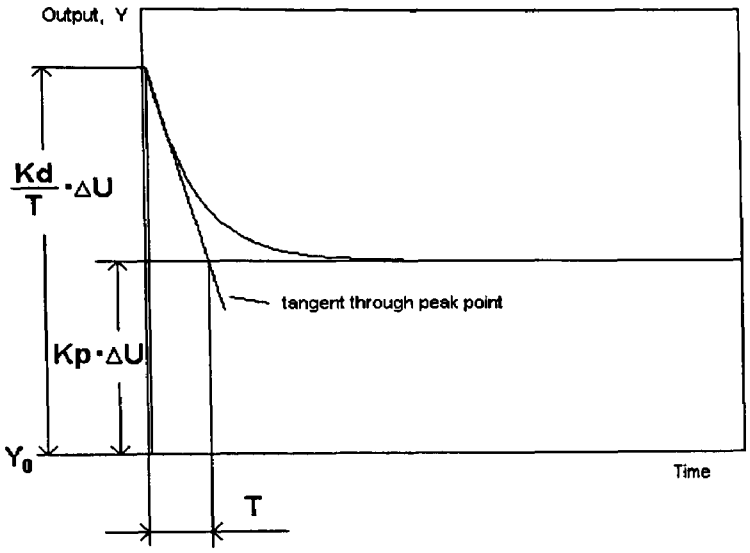
$$D \approx -\frac{\ln \frac{A2}{A1}}{\sqrt{\pi^2 + \left(\ln \frac{A2}{A1}\right)^2}}$$

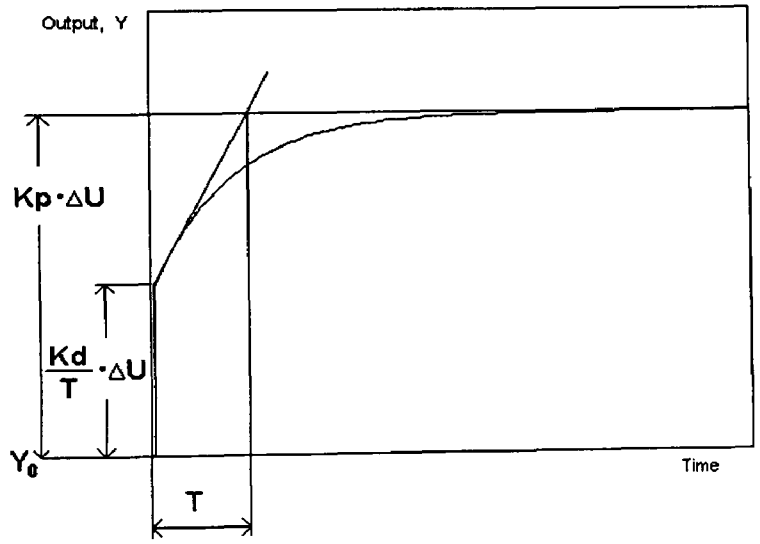
$$T \approx \frac{(Ts/2)}{\pi}\sqrt{1 - D^2}$$

Figure A6-8: "PIT2" Identification

$$Kp \approx \frac{A1}{\Delta U}$$

PDT1:





$$G(s) = \frac{Kp \cdot (1 + Tv \cdot s)}{1 + T \cdot s}$$

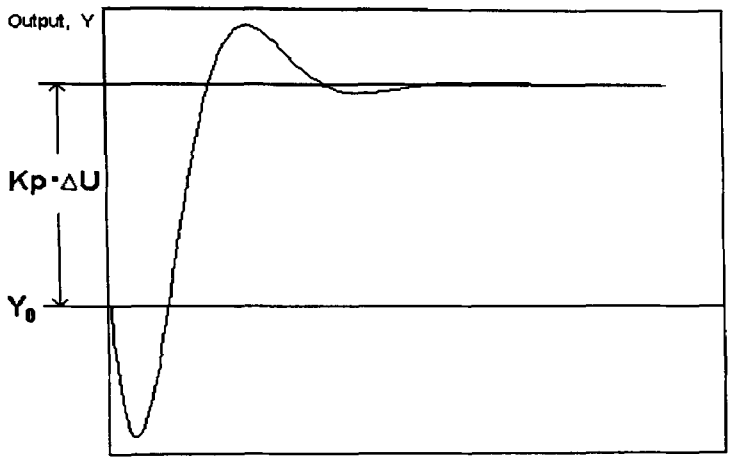
$$Kp = \frac{Y_\infty - Y_0}{\Delta U} = \frac{(Kp \cdot \Delta U)}{\Delta U}$$

$$Kd = \left(\frac{Kd}{T} \cdot \Delta U \right) \cdot \frac{T}{\Delta U}$$

$$Tv = \frac{Kd}{Kp}$$

Figure A6-9: "PDT1" Identification

PDT2:



$$G(s) = \frac{Kp \cdot (1 + Tv \cdot s)}{1 + 2DT \cdot s + T^2 \cdot s^2}$$

$$Kp = \frac{Y_\infty - Y_0}{\Delta U} = \frac{(Kp \cdot \Delta U)}{\Delta U}$$

(No simplified identification of the parameters *D*, *T* and *Tv* for PDT2 implemented!)

Figure A6-10: "PDT2" Identification

The Generated ASCII-Report "FUNNELX.M"

```
echo on
% >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
% STRUCTURED SUMMARY OF THE PROCESS INFORMATION
% ACQUIRED IN MODELing
%
% by typing 'funnelx' in the MATLAB workspace, numerical
% information is directly passed to MATLAB; further data can be
% transferred by typing 'funnela'
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
% NOTE: 'NULL' and 'void' are equivalent to blanks - i.e., no answer
%       has been given!
%
% ------------------------------------------------------------------------------
%                        COMPONENT information:
% ------------------------------------------------------------------------------
% Component Name: funnel
% Process Domain: ProcessEng
% Process Type: Tank
% known Sub-Components:
%
% ---------- Model Documentation: ----------
% Version No.: 1
% Purpose of Model: MODELing and fuzTF application test
% Assumptions:
% Internal Verification and External Validation Tests:
% Results of Model Tests:
% Comments:
%
% ------------------------------------------------------------------------------
%                        TERMINAL information:
% ------------------------------------------------------------------------------
% Number of Inputs: 2
% Number of Outputs: 1
% Number of Disturbances: 0
% Input Variables: Area Qin
% Output Variables: WaterH
% Disturbance Variables:
%
% ----------Input 'Area' ----------
% Full Name of Variable: tank outlet area
% Symbol: A
% Units: m^2
% Medium: NULL
% Acting as MAIN Input on the Outputs:
% Nonlinear Influence on the Outputs: WaterH
% Operating Points: .05 .025 .035
% Operating Range: 0 to .05
% Max. Rate of Change: NULL
% Min. Alarm: NULL
% Max. Alarm: NULL
% Lower Tolerance: NULL
% Upper Tolerance: NULL
```

```
% Actuator Type: NULL
% Actuator Properties:
% Transducer Type: NULL
% Transducer Properties:
% Signal directly Measurable and Accessible: Yes
% Possibility of Staircase Input Signal: Yes
% Possibility of Step Input Signal: Yes
% Possibility of Impulse Input Signal: Yes
% Possibility of sequential Impulse Signal: Yes
%
% ------------------Input 'Qin' ------------------------------------------------------------------------
% Full Name of Variable: water flow into the tank
% Symbol: Qin
% Units: m^3/s
% Medium: water
% Acting as MAIN Input on the Outputs: WaterH
% Nonlinear Influence on the Outputs:
% Operating Points: .05 .08 .11 .15 .23 .28
% Operating Range: 0 to .3
% Max. Rate of Change: NULL
% Min. Alarm: NULL
% Max. Alarm: NULL
% Lower Tolerance: NULL
% Upper Tolerance: NULL
% Actuator Type: NULL
% Actuator Properties:
% Transducer Type: NULL
% Transducer Properties:
% Signal directly Measurable and Accessible: Yes
% Possibility of Staircase Input Signal: Yes
% Possibility of Step Input Signal: Yes
% Possibility of Impulse Input Signal: Yes
% Possibility of sequential Impulse Signal: Yes
%
% ------------------Output 'WaterH' ------------------------------------------------------------------
% Full Name of Variable: water level in the funnel tank
% Symbol: h
% Units: m
% Medium: NULL
% The MAIN Input on WaterH is: Qin
% Ranking of nonlinear Influences according to Importance for WaterH : WaterH Area
% Operating Points: .131 1.616 1.094 .566
% Operating Range: 0 to 2
% Max. Rate of Change: NULL
% Min. Alarm: 0
% Max. Alarm: 1.9
% Lower Tolerance: .1
% Upper Tolerance: 1.8
% Transducer Type: NULL
% Transducer Properties:
% Signal directly Measurable and Accessible: Yes
%
%
```

```
% ------------------------------------------------------------------------------
%                        MODEL information:
% ------------------------------------------------------------------------------
% In general, multiple input / multiple output (MIMO) components are
% considered. The following information is grouped around the individual
% outputs, because the overall MIMO process is split into MISO parts which
% in turn are based on SISO relationships.
% ------------------------------------------------------------------------------
%                 Dynamic Mathematical Descriptions
% ------------------------------------------------------------------------------
% The dynamic behaviour is described for each output by a continuous transfer
% function (TF) in 's', relating the MAIN input (cf. above, section 'Output
% Terminals') to the considered output. Below, the numerators and
% denominators of these transfer functions are specified separately in MATLAB
% terminology, in terms of the coefficients of decreasing orders of s.
% [a3 a2 a1 a0] refers for example to the term a3*s^3+a2*s^2+a1*s+a0 .
% These SISO transfer functions can accomodate additional influences on each
% output (-> MISO) through variations of the parameters (e.g. a3,a2,a1,a0).
% If the transfer function varies even in its order, the appropriate parameters
% are conditionally set to 0. The shown numerators and denominators are
% therefore 'Common' numerators and denominators which accommodate all orders
% that have been derived per output.
% The numerator is always normalised by the static gain, b0, to allow for its
% separate consideration in the static characteristic.
% For the direct application of the transfer function structures in Simulink
% please refer to the blockdiagram 'identi.m'!
%
%
% ------------------TF, relating Input 'Qin' to Output 'WaterH': ----------------------------------

NUM_WaterH=[1];

DEN_WaterH=[a1 a0];

% The parameters of the transfer functions (ai, bi) vary possibly, depending
% on the settings of up to two additional influences. Therefore, these parameters
% are defined as matrices (one matrix for each parameter), with the different rows
% showing the variation of the parameter due to the 1st influence and the
% differences between the columns resulting from the 2nd influence. Each row and
% each column is related to one of the previously specified levels of influence 1
% and influence 2, respectively, with increasing row and column numbers referring
% to increasing influence levels.
% Since these matrices are defined in the simulation code interface as Rule Base
% matrices (prefix 'RB'), these are not repeated here. By typing
% 'funnela', this data is added to the MATLAB workspace.
% RBa1_... is therefore the matrix that describes the variations of parameter 'a1'
% due to the influences. The extension (...) of the matrix name defines exactly
% the component and output it refers to. The pre-defined influence levels that
% are associated with the matrix positions are given as arguments of the
% function 'membersf()'.
%
% Dead Times and Rate Limiters with possibly varying parameters are likewise
% defined as Rule Base matrices: RBTd_... and RBRateLim_..., respectively.
%
%
```

```
% ------------------------------------------------------------------------------------------
%                  Nonlinear Static Characteristics
% ------------------------------------------------------------------------------------------
% The following static characteristics ('SC') are defined by lookup-tables
% in two-column format, with the input levels, U (MAIN input!), in the left
% column and the associated steady state output levels, Y, in the right
% column. Multiple dimensional Characteristics are represented by a collection
% of such 2-dimensional SC's, each representing a 'cut' through the N-
% dimensional surface.
%
% The first set of static characteristics ('SC_...') results from the dynamic
% modelling approach, whereas the second set ('SC2_...') is derived from purely
% static modelling. The results SC_... and SC2_... overlap or complement
% each other.
%
% The Labels of the static characteristic lookup tables always commence
% with 'SC_' (or 'SC2_'), followed by the name of the output and a simple,
% meaningless number to distinguish between the different settings of the
% additional influences.
%
% ----------------------SC_... for Output 'WaterH': ----------------------------------------
% The static relationship between the input 'Qin' and the output 'WaterH'
% is nonlinear, due to the 1st Influence ('WaterH').
%
% For influence-setting Area = .025 :
SC_WaterH_1=[.04 .131
             .0833 .566
             .1158 1.094
             .1407 1.616
             ];
% For influence-setting Area = .035 :
SC_WaterH_2=[.056 .131
             .1166 .566
             .1621 1.094
             .1970 1.616
             ];
% For influence-setting Area = .05 :
SC_WaterH_3=[.08 .131
             .1665 .566
             .2315 1.094
             .2814 1.616
             ];
%
%
% ---------- ALTERNATIVE static characteristics ... ----------------------------------
%
% Second set of lookup tables (SC2_...), generated from the purely
% static characteristic modelling approach (NLSC class).
%
% In case that a characteristic defined above is featured here again,
% the following data can be compared with the previously defined lookup tables which
% were extracted from dynamic system modelling (Math class).
%
%
```

```
% ----------------------SC2_... for Output 'WaterH': ----------------------------------------------
% For setting of influence 'Area' = .025:
SC2_WaterH_1=[.05 .23
              .1 .83
              .15 2
              ];
%
% For setting of influence 'Area' = .035:
SC2_WaterH_2=[.05 .1
              .1 .4
              .15 .95
              ];
%
% For setting of influence 'Area' = .05:
SC2_WaterH_3=[.05 .08
              .1 .23
              .15 .45
              ];
%
%
%
% ------------------------------------------------------------------------------------------------
%                        Production Rules
% ------------------------------------------------------------------------------------------------
% No Production Rules defined!
%
%
%
% ------------------------------------------------------------------------------------------------
%                    Attributes, describing the component
% ------------------------------------------------------------------------------------------------
%
% Remark: as before, the considerations per output refer to its relationship
% with the associated MAIN input. Additional influences (= further inputs) are
% taken account of.
%
% ------------------------------------------------------------------------------------------------
%
% ----- Linear Dynamics Attributes for the Relation 'Qin' to Output 'WaterH': ------------------
%
% Does the output...
% ...settle to a steady state?    Yes
% ...overshoot after a step input?    No
% ...oscillate after a step input?    No
% ...respond without lag?    No
% ...respond with increasing speed?    No
% ...respond with decreasing speed?    Yes
% ...show non-minimal phase behaviour?    No
%
%
% ----- Nonlinear Dynamics Attributes for the Relation 'Qin' to Output 'WaterH': ------------
%
% Does the output...
% ...show nonlinear dynamic behaviour?    Yes
% ...have a limited rate of change?    No
```

```
% ...have a limited rate of change that varies?    No
% ...react after a dead time?    No
% ...react after a dead time that varies?    No
% ...behaviour indicate a variant time constant?    Yes
% ...behaviour indicate a variant damping ratio?    void
%
%
% ----- Nonlinear Statics Attributes for the Relation 'Qin' to Output 'WaterH': ----------
%
% ---------- For setting of influence 'Area' = .025:----------
%
% Is the static relationship nonlinear?    Yes
% Nonlinearity of 'Hammerstein' Structure?    No
% Nonlinearity of 'Wiener' Structure?    Yes
% Multiple Valued relationship (depending on direction of input change) ?    No
% Multiple Variable relationship (depending on an additional influence)?    Yes
% Time variant relationship?    No
% Type of time variation?    void
% Relevant time scale?    void
% Discontinuous relationship?    No
% Discontinuity type?    void
% Static characteristic with positive slope?    Yes
% Positive slope data:
SlopePos_WaterH=[.05 .23
                 .1 .83
                 .15 2. ];
%
% Static characteristic with negative slope?    No
% Static characteristic with upper saturation?    No
% Static characteristic with lower saturation?    No
% Static characteristic with maxima?    No
% Number of maxima?    void
% Static characteristic with minima?    Yes
% Number of minima?    1
% Data of minima in the characteristic:
Minima_WaterH=[0.025 0 ];
%
% Static characteristic with intermediate dead zones?    No
% Number of dead zones?    void

% ---------- For setting of influence 'Area' = .035:----------
%
% Is the static relationship nonlinear?    Yes
% Nonlinearity of 'Hammerstein' Structure?    No
% Nonlinearity of 'Wiener' Structure?    Yes
% Multiple Valued relationship (depending on direction of input change) ?    No
% Multiple Variable relationship (depending on an additional influence)?    Yes
% Time variant relationship?    No
% Type of time variation?    void
% Relevant time scale?    void
% Discontinuous relationship?    No
% Discontinuity type?    void
% Static characteristic with positive slope?    Yes
```

```
% Positive slope data:
SlopePos_WaterH=[.05 .1
                 .1 .4
                 .15 .95];
%
% Static characteristic with negative slope?    No
% Static characteristic with upper saturation?    No
% Static characteristic with lower saturation?    No
% Static characteristic with maxima?    No
% Number of maxima?    void
% Static characteristic with minima?    Yes
% Number of minima?    1
% Data of minima in the characteristic:
Minima_WaterH=[0.03 0];
%
% Static characteristic with intermediate dead zones?    No
% Number of dead zones?    void
%
% ------------------For setting of influence 'Area' = .050:----------------------------------
%
% Is the static relationship nonlinear?    Yes
% Nonlinearity of 'Hammerstein' Structure?    No
% Nonlinearity of 'Wiener' Structure?    Yes
% Multiple Valued relationship (depending on direction of input change) ?    No
% Multiple Variable relationship (depending on an additional influence)?    Yes
% Time variant relationship?    No
% Type of time variation?    void
% Relevant time scale?    void
% Discontinuous relationship?    No
% Discontinuity type?    void
% Static characteristic with positive slope?    Yes
% Positive slope data:
SlopePos_WaterH=[.05 .08
                 .1 .23
                 .15 .45 ];
%
% Static characteristic with negative slope?    No
% Static characteristic with upper saturation?    No
% Static characteristic with lower saturation?    No
% Static characteristic with maxima?    No
% Number of maxima?    void
% Static characteristic with minima?    Yes
% Number of minima?    1
% Data of minima in the characteristic:
Minima_WaterH=[0.04 0 ];
%
% Static characteristic with intermediate dead zones?    No
% Number of dead zones?    void
%
% ##########################################################################
% ATTENTION! To pass more data - especially for fuzTF simulation - to MATLAB,
% type 'funnela' !
% ##########################################################################
echo off
identi      %  /* ..to call the Simulink blocks for identification */
```

The Generated MATLAB Data File "FUNNELA.M"

```
% >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
% INTERFACE FILE WITH PARAMETER INFORMATION FOR USE IN MATLAB;
% GENERATED IN MODELing
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
% -----------------------------------------------------------------------------------------------
%    Section of simulation parameters for fuzSC / fuzTF simulation:
% -----------------------------------------------------------------------------------------------
% -----------------------------------------------------------------------------------------------
% All following data refers to the MISO subcomponent related to output WaterH:
% -----------------------------------------------------------------------------------------------
% The Rule-Base matrix for the fuzSC-offset Yo
% (related to the output WaterH):

RBYo_funnelWaterH=[-0.097 -0.1098 -0.117
                   -0.50024 -0.51838 -0.5329
                   -1.00198 -1.02951 -1.0821
                   -1.49347 -1.536 -1.64824
                   ];

% The Rule-Base matrix for the polynomial transfer function parameter a0
% (related to the output WaterH):

RBa0_funnelWaterH=[1 1 1
                   1 1 1
                   1 1 1
                   1 1 1
                   ];

% The Rule-Base matrix for the polynomial transfer function parameter a1
% (related to the output WaterH):

RBa1_funnelWaterH=[.39 .36 .32
                   2.62 2.29 1.88
                   8.47 7.08 6.34
                   20.14 16.28 12.87
                   ];

% The Rule-Base matrix for the polynomial transfer function parameter b0
% (related to the output WaterH):

RBb0_funnelWaterH=[5.7 4.3 3.1
                   12.8 9.3 6.6
                   18.1 13.1 9.4
                   22.1 16 11.6
                   ];

%
% The Rule-Base matrix for the dead time Td
% (related to the output WaterH):
%
% (only for information on possibly varying dead times - cannot be simulated in MATLAB!)
```

```
RBTd_funnelWaterH=[0 0 0
                   0 0 0
                   0 0 0
                   0 0 0
                   ];

% The Rule-Base matrix for the Rate Limiter
% (related to the output WaterH):
%
% (only for information on possibly existing and varying Rate Limiters - cannot be
% simulated in MATLAB!)

RBRateLim_funnelWaterH=[inf inf inf
                        inf inf inf
                        inf inf inf
                        inf inf inf
                        ];

% To generate the membership function for Influence 1
% related to the output WaterH:
% (The arguments of function 'membersf()' are the levels of 'WaterH'
% that refer to the rows in the Rule Base matrices of 'WaterH'!)

MF1_funnelWaterH=membersf(.131, .566, 1.094, 1.616)
xlabel('MF1_funnelWaterH')
pause

% To generate the membership function for Influence 2
% related to the output WaterH:
% (The arguments of function 'membersf()' are the levels of 'Area'
% that refer to the columns in the Rule Base matrices of 'WaterH'!)

MF2_funnelWaterH=membersf(.025, .035, .05)
xlabel('MF2_funnelWaterH')
pause

% ------------------------------------------------------------------------------------------
% Parameters for alternative static characteristic, generated from
% the interactive nonlinear static characteristic modelling approach.
% This fuzSC2-based static characteristic can be tested as a replacement for fuzSC.
% ------------------------------------------------------------------------------------------
% The Rule-Base matrix 'Yo_2 ..' for fuzSC2
% (related to the output WaterH):

RBYo_2funnelWaterH=[.23 .1 .08
                    .83 .4 .23
                    2 .95 .45
                    ];

% To generate the membership function for fuzSC2-input (=Influence 1)
% (related to the output WaterH):

MF1_2funnelWaterH=membersf(.05, .1, .15)
xlabel('MF1_2funnelWaterH')
pause
```

```
% To generate the membership function for fuzSC2 Influence 2
% (related to the output WaterH):

MF2_2funnelWaterH=membersf(.025, .035, .05)
xlabel('MF2_2funnelWaterH')
pause

% ------------------------------------------------------------------------------
% ------------------------------------------------------------------------------
% All matices are automatically saved as .mat file:

save  funnel

% ------------------------------------------------------------------------------
% ------------------------------------------------------------------------------
% call blockdiagrams:

funnelb
funnel
whos
```

The Generated Simulink Blockdiagram File "FUNNEL.M"

```
function [ret,x0,str,ts,xts]= funnel (t,x,u,flag);
% >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
% fuzTF BLOCKDIAGRAM FOR USE IN MATLAB IN CONJUNCTION WITH fuztfsim.m;
% GENERATED IN MODELing
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
% funnel  is the M-file description of the SIMULINK system named  funnel .
%      The block-diagram can be displayed by typing:  funnel .
%
%        SYS= funnel (T,X,U,FLAG) returns depending on FLAG certain
%        system values given time point, T, current state vector, X,
%        and input vector, U.
%        FLAG is used to indicate the type of output to be returned in SYS.
%
%        Setting FLAG=1 causes  funnel  to return state derivatives, FLAG=2
%        discrete states, FLAG=3 system outputs and FLAG=4 next sample
%        time. For more information and other options see SFUNC.
%
%        Calling  funnel  with a FLAG of zero:
%        [SIZES]= funnel ([],[],[],0),  returns a vector, SIZES, which
%        contains the sizes of the state vector and other parameters.
%                SIZES(1) number of states
%                SIZES(2) number of discrete states
%                SIZES(3) number of outputs
%                SIZES(4) number of inputs
%                SIZES(5) number of roots (currently unsupported)
%                SIZES(6) direct feedthrough flag
```

```
%               SIZES(7) number of sample times
%
%       For the definition of other parameters in SIZES, see SFUNC.
%       See also, TRIM, LINMOD, LINSIM, EULER, RK23, RK45, ADAMS, GEAR.
%
% Note: This M-file is only used for saving graphical information;
%       after the model is loaded into memory an internal model
%       representation is used.
%
% the system will take on the name of this mfile:
sys = mfilename;
new_system(sys)
simver(1.3)
if (0 == (nargin + nargout))
   set_param(sys,'Location',[10,40,620,410])
   open_system(sys)
end;
set_param(sys,'algorithm',    'RK-45')
set_param(sys,'Start time',   '0.0')
set_param(sys,'Stop time',    '18')
set_param(sys,'Min step size', '0.0001')
set_param(sys,'Max step size', '0.10')
set_param(sys,'Relative error','1e-3')
set_param(sys,'Return vars',  '')

% ------------------------------------------------------------------------------
% The following blocks refer to the MISO subcomponent related to output WaterH:
% ------------------------------------------------------------------------------
% ------------------------------------------------------------------------------
%               fuzTF and fuzSC blocks for WaterH:
% ------------------------------------------------------------------------------
add_block('built-in/Transfer Fcn',[sys,'/','fuzTF_funnelWaterH'])
set_param([sys,'/','fuzTF_funnelWaterH'],...
                  'Numerator','[1]',...
                  'Denominator','[a1_funnelWaterH a0_funnelWaterH ]',...
                  'position',[255, 20 ,525, 70 ])
add_block('built-in/Note',[sys,'/',['Please connect the blocks IN1..',13,'and IN2.. appropriately to
define',13,'the influences 1 and 2!']])
set_param([sys,'/',['Please connect the blocks IN1..',13,'and IN2.. appropriately to define',13,'the
influences 1 and 2!']],...
                  'position',[525,95,530,100])
add_block('built-in/Fcn',[sys,'/','fuzSC_funnelWaterH'])
set_param([sys,'/','fuzSC_funnelWaterH'],...
                  'Expr','b0_funnelWaterH*(u[1])+Yo_funnelWaterH',...
                  'position',[100, 35 ,205, 55 ])
add_block('built-in/To Workspace',[sys,'/','Influence1 on WaterH',13,' of funnel'])
set_param([sys,'/','Influence1 on WaterH',13,' of funnel'],...
                  'mat-name','IN1_funnelWaterH',...
                  'buffer','1',...
                  'position',[170, 90 ,280, 110 ])
add_block('built-in/To Workspace',[sys,'/','Influence2 on WaterH',13,' of funnel'])
set_param([sys,'/','Influence2 on WaterH',13,' of funnel'],...
                  'mat-name','IN2_funnelWaterH',...
                  'buffer','1',...
                  'position',[315, 90 ,425, 110 ])
```

```
add_block('built-in/Outport',[sys,'/','Y(:,2), Qin',13,'data record'])
set_param([sys,'/','Y(:,2), Qin',13,'data record'],...
                'Port','2',...
                'position',[105,85,125,105])
add_block('built-in/Outport',[sys,'/','Y(:,1), WaterH',13,'data record'])
set_param([sys,'/','Y(:,1), WaterH',13,'data record'],...
                'Port','1',...
                'position',[560,35,580,55])
add_line(sys,[55,45;95,45])
add_line(sys,[55,45;70,45;70,95;100,95])
add_line(sys,[210,45;250,45])
add_line(sys,[530,45;555,45])
add_block('built-in/Note',[sys,'/',[' 1) Please',13,'connect',13,'input Qin',13,'to a',13,' source!']])
set_param([sys,'/',[' 1) Please',13,'connect',13,'input Qin',13,'to a',13,' source!']],...
                'position',[35,45,40,50])

drawnow

% Return any arguments.
if (nargin | nargout)
        % Must use feval here to access system in memory
        if (nargin > 3)
                if (flag == 0)
                        eval(['[ret,x0,str,ts,xts]=',sys,'(t,x,u,flag);'])
                else
                        eval(['ret =', sys,'(t,x,u,flag);'])
                end
        else
                [ret,x0,str,ts,xts] = feval(sys);
        end
else
        drawnow % Flash up the model and execute load callback
end
```

The *MODEL*^{ing} -Internal Representation Of The Funnel Model As A Collection Of Instances With Their Slots In KAL-Code: FUNNEL.KAL

The following funnel.kal file was automatically saved from within the *MODEL*^{ing} program (button 'Save') and can be retrieved in later modelling sessions via the browser.

```
/*************************************
 **** INSTANCE: funnel
 *************************************/
MakeInstance( funnel, Component );
SetSlotOption( funnel:ListOutputs, MULTIPLE );
SetValue( funnel:ListOutputs, WaterH );
SetSlotOption( funnel:ListInputs, MULTIPLE );
SetValue( funnel:ListInputs, Area, Qin );
```

```
SetSlotOption( funnel:ListDisturbances, MULTIPLE );
ClearList( funnel:ListDisturbances );
funnel:NoOutputs = 1;
SetSlotOption( funnel:NoOutputs, IMAGE, Edit19 );
funnel:NoInputs = 2;
SetSlotOption( funnel:NoInputs, IMAGE, Edit18 );
funnel:NoDisturbances = 0;
SetSlotOption( funnel:NoDisturbances, IMAGE, Edit20 );
funnel:ProcessDomain = ProcessEng;
SetSlotOption( funnel:ProcessDomain, IMAGE, ComboBox3 );
funnel:ProcessType = Tank;
SetSlotOption( funnel:ProcessType, IMAGE, ComboBox2 );
SetValue( funnel:ListTerminals, Area, Qin, WaterH );

    /**************************************
     **** INSTANCE: funnelDOC
     **************************************/
MakeInstance( funnelDOC, Documentation );
SetValue( funnelDOC:ModelPurpose, MODELing, and, fuzTF, application, test );
funnelDOC:VersionNo = 1;

    /**************************************
     **** INSTANCE: Area
     **************************************/
MakeInstance( Area, Input );
Area:Symbol = A;
Area:Units = "m^2";
SetValue( Area:OperatingPoints, .05, .025, .035 );
Area:OperatingMax = .05;
Area:OperatingMin = 0;
Area:Measurable = Yes;
Area:StaircaseSignalOK = Yes;
Area:StepSignalOK = Yes;
Area:ImpulseSignalOK = Yes;
Area:PRBSSignalOK = Yes;
SetValue( Area:InfluenceOn, WaterH );
Area:Name = "tank outlet area";

    /**************************************
     **** INSTANCE: Qin
     **************************************/
MakeInstance( Qin, Input );
Qin:Symbol = Qin;
Qin:Units = "m^3/s";
SetValue( Qin:OperatingPoints, .05, .08, .11, .15, .23, .28 );
Qin:OperatingMax = .3;
Qin:OperatingMin = 0;
Qin:Medium = water;
Qin:Measurable = Yes;
Qin:StaircaseSignalOK = Yes;
Qin:StepSignalOK = Yes;
Qin:ImpulseSignalOK = Yes;
Qin:PRBSSignalOK = Yes;
SetValue( Qin:MainInputFor, WaterH );
Qin:Name = "water flow into the tank";
```

```
/*************************************
 **** INSTANCE: WaterH
 *************************************/
MakeInstance( WaterH, Output );
WaterH:Symbol = h;
WaterH:Units = m;
SetValue( WaterH:OperatingPoints, .131, 1.616, 1.094, .566 );
WaterH:OperatingMax = 2;
WaterH:OperatingMin = 0;
WaterH:MaxAlarm = 1.9;
WaterH:MinAlarm = 0;
WaterH:Measurable = Yes;
SetValue( WaterH:MathIn1_d, .131, .566, 1.094, 1.616 );
SetValue( WaterH:ModelledInflSett, ".131 - .025", ".131 - .035", ".131 - .05", "1.094 - .025", "1.094 -
.05", ".566 - .025", ".566 - .05", "1.616 - .05", "1.616 - .035", "1.616 - .025", ".566 - .035", "1.094 -
.035", ".131 - .025", "1.094 - .025" );
WaterH:MainInput = Qin;
SetSlotOption( WaterH:InfluencesRanking, ALLOWABLE_VALUES, Area, Qin, WaterH );
SetValue( WaterH:InfluencesRanking, WaterH, Area );
SetValue( WaterH:MathInfluence1_data, .131, .566, 1.094, 1.616 );
WaterH:Influence1Math = WaterH;
WaterH:Influence2Math = Area;
SetValue( WaterH:InfluenceOn, WaterH );
SetSlotOption( WaterH:In1MathSetting, ALLOWABLE_VALUES, .131, .566, 1.094, 1.616 );
WaterH:In1MathSetting = .131;
SetSlotOption( WaterH:In1MathSetting, IMAGE, RadioButtonGroup1 );
SetSlotOption( WaterH:In2MathSetting, ALLOWABLE_VALUES, .025, .035, .05 );
WaterH:In2MathSetting = .025;
SetSlotOption( WaterH:In2MathSetting, IMAGE, RadioButtonGroup2 );
WaterH:In2MathRefNo_x = 2;
WaterH:NumberModelInstances = 12;
SetValue( WaterH:MathInstances_1, WaterHH3, WaterHH7, WaterHH5, WaterHH8 );
SetValue( WaterH:MathInstances_2, WaterHH1, WaterHH6, WaterHH4, WaterHH10 );
SetValue( WaterH:MathInstances_3, WaterHH2, WaterHH11, WaterHH12, WaterHH9 );
SetValue( WaterH:MathInstances_4, dummy, dummy, dummy, dummy );
SetValue( WaterH:MathInstances_5, dummy, dummy, dummy, dummy );
SetValue( WaterH:MathInstances_6, dummy, dummy, dummy, dummy );
SetValue( WaterH:MathInstances_7, dummy, dummy, dummy, dummy );
SetValue( WaterH:MathInstances_8, dummy, dummy, dummy, dummy );
SetValue( WaterH:MathInstances_9, dummy, dummy, dummy, dummy );
WaterH:MaxOrderDenominator = 1;
WaterH:CommonDEN = "[a1 a0]";
SetValue( WaterH:NLsAttribInfluence2, .025, .035, .050 );
SetValue( WaterH:NLsAttribInstances, WaterHH13, WaterHH19, WaterHH20 );
WaterH:Influence2NLSC = Area;
WaterH:Influence2Attrib = Area;
WaterH:NLdAttribInstance = WaterHH14;
WaterH:LinAttribInstance = WaterHH15;
WaterH:OpenLoop? = Yes;
SetValue( WaterH:NLSCInfluence2, .025, .035, .05 );
SetValue( WaterH:NLSCInstances, WaterHH18, WaterHH17, WaterHH16 );
SetValue( WaterH:NLSCInDataU, .05, .1, .15 );
WaterH:UpperTolerance = 1.8;
WaterH:LowerTolerance = .1;
```

```
SetValue( WaterH:MathInfluence2_data, .025, .035, .05 );
SetValue( WaterH:MathInfluence2_Order, 2, 3, 1 );
ClearList( WaterH:RemainingInflSett );
WaterH:Name = "water level in the funnel tank";

    /*************************************
     **** INSTANCE: WaterHH16
     *************************************/
MakeInstance( WaterHH16, NLSC );
SetValue( WaterHH16:OutDataY, .08, .23, .45 );

    /*************************************
     **** INSTANCE: WaterHH17
     *************************************/
MakeInstance( WaterHH17, NLSC );
SetValue( WaterHH17:OutDataY, .1, .4, .95 );

    /*************************************
     **** INSTANCE: WaterHH18
     *************************************/
MakeInstance( WaterHH18, NLSC );
SetValue( WaterHH18:OutDataY, .23, .83, 2 );

    /*************************************
     **** INSTANCE: WaterHH2
     *************************************/
MakeInstance( WaterHH2, PT1 );
WaterHH2:OutputInstance = WaterH;
WaterHH2:OperatingPointIn = .056;
WaterHH2:OperatingPointOut = .131;
WaterHH2:DeadTimeInfo = N;
WaterHH2:Td = 0;
WaterHH2:ComplexPoles = No;
WaterHH2:Numerator = Kp;
WaterHH2:Denominator = "1 + T s";
WaterHH2:InputStep = .01;
WaterHH2:KpX = .043;
WaterHH2:T = .36;
WaterHH2:Kp = 4.3;
WaterHH2:b0 = 4.3;
WaterHH2:a1 = .36;
WaterHH2:Yo = -0.1098;

    /*************************************
     **** INSTANCE: WaterHH3
     *************************************/
MakeInstance( WaterHH3, PT1 );
WaterHH3:OutputInstance = WaterH;
WaterHH3:OperatingPointIn = .08;
WaterHH3:OperatingPointOut = .131;
WaterHH3:DeadTimeInfo = N;
WaterHH3:Td = 0;
WaterHH3:ComplexPoles = No;
WaterHH3:Numerator = Kp;
WaterHH3:Denominator = "1 + T s";
```

```
WaterHH3:InputStep = .01;
WaterHH3:KpX = .031;
WaterHH3:T = .32;
WaterHH3:Kp = 3.1;
WaterHH3:b0 = 3.1;
WaterHH3:a1 = .32;
WaterHH3:Yo = -0.117;

    /*************************************
     **** INSTANCE: WaterHH5
     *************************************/
MakeInstance( WaterHH5, PT1 );
WaterHH5:OutputInstance = WaterH;
WaterHH5:OperatingPointIn = .2315;
WaterHH5:OperatingPointOut = 1.094;
WaterHH5:DeadTimeInfo = N;
WaterHH5:Td = 0;
WaterHH5:ComplexPoles = No;
WaterHH5:Numerator = Kp;
WaterHH5:Denominator = "1 + T s";
WaterHH5:InputStep = .01;
WaterHH5:KpX = .094;
WaterHH5:T = 6.34;
WaterHH5:Kp = 9.4;
WaterHH5:b0 = 9.4;
WaterHH5:a1 = 6.34;
WaterHH5:Yo = -1.0821;

    /*************************************
     **** INSTANCE: WaterHH6
     *************************************/
MakeInstance( WaterHH6, PT1 );
WaterHH6:OutputInstance = WaterH;
WaterHH6:OperatingPointIn = .0833;
WaterHH6:OperatingPointOut = .566;
WaterHH6:DeadTimeInfo = N;
WaterHH6:Td = 0;
WaterHH6:ComplexPoles = No;
WaterHH6:Numerator = Kp;
WaterHH6:Denominator = "1 + T s";
WaterHH6:InputStep = .01;
WaterHH6:KpX = .128;
WaterHH6:T = 2.62;
WaterHH6:Kp = 12.8;
WaterHH6:b0 = 12.8;
WaterHH6:a1 = 2.62;
WaterHH6:Yo = -0.50024;

    /*************************************
     **** INSTANCE: WaterHH7
     *************************************/
MakeInstance( WaterHH7, PT1 );
WaterHH7:OutputInstance = WaterH;
WaterHH7:OperatingPointIn = .1665;
WaterHH7:OperatingPointOut = .566;
```

```
WaterHH7:DeadTimeInfo = N;
WaterHH7:Td = 0;
WaterHH7:ComplexPoles = No;
WaterHH7:Numerator = Kp;
WaterHH7:Denominator = "1 + T s";
WaterHH7:InputStep = .01;
WaterHH7:KpX = .066;
WaterHH7:T = 1.88;
WaterHH7:Kp = 6.6;
WaterHH7:b0 = 6.6;
WaterHH7:a1 = 1.88;
WaterHH7:Yo = -0.5329;

    /*************************************
     **** INSTANCE: WaterHH8
     *************************************/
MakeInstance( WaterHH8, PT1 );
WaterHH8:OutputInstance = WaterH;
WaterHH8:OperatingPointIn = .2814;
WaterHH8:OperatingPointOut = 1.616;
WaterHH8:DeadTimeInfo = N;
WaterHH8:Td = 0;
WaterHH8:ComplexPoles = No;
WaterHH8:Numerator = Kp;
WaterHH8:Denominator = "1 + T s";
WaterHH8:InputStep = .01;
WaterHH8:KpX = .116;
WaterHH8:T = 12.87;
WaterHH8:Kp = 11.6;
WaterHH8:b0 = 11.6;
WaterHH8:a1 = 12.87;
WaterHH8:Yo = -1.64824;

    /*************************************
     **** INSTANCE: WaterHH9
     *************************************/
MakeInstance( WaterHH9, PT1 );
WaterHH9:OutputInstance = WaterH;
WaterHH9:OperatingPointIn = .1970;
WaterHH9:OperatingPointOut = 1.616;
WaterHH9:DeadTimeInfo = N;
WaterHH9:Td = 0;
WaterHH9:ComplexPoles = No;
WaterHH9:Numerator = Kp;
WaterHH9:Denominator = "1 + T s";
WaterHH9:InputStep = .01;
WaterHH9:KpX = .16;
WaterHH9:T = 16.28;
WaterHH9:Kp = 16;
WaterHH9:b0 = 16;
WaterHH9:a1 = 16.28;
WaterHH9:Yo = -1.536;
```

```
    /*************************************
     **** INSTANCE: WaterHH10
     *************************************/
MakeInstance( WaterHH10, PT1 );
WaterHH10:OutputInstance = WaterH;
WaterHH10:OperatingPointIn = .1407;
WaterHH10:OperatingPointOut = 1.616;
WaterHH10:DeadTimeInfo = N;
WaterHH10:Td = 0;
WaterHH10:ComplexPoles = No;
WaterHH10:Numerator = Kp;
WaterHH10:Denominator = "1 + T s";
WaterHH10:InputStep = .01;
WaterHH10:KpX = .221;
WaterHH10:T = 20.14;
WaterHH10:Kp = 22.1;
WaterHH10:b0 = 22.1;
WaterHH10:a1 = 20.14;
WaterHH10:Yo = -1.49347;

    /*************************************
     **** INSTANCE: WaterHH11
     *************************************/
MakeInstance( WaterHH11, PT1 );
WaterHH11:OutputInstance = WaterH;
WaterHH11:OperatingPointIn = .1166;
WaterHH11:OperatingPointOut = .566;
WaterHH11:DeadTimeInfo = N;
WaterHH11:Td = 0;
WaterHH11:ComplexPoles = No;
WaterHH11:Numerator = Kp;
WaterHH11:Denominator = "1 + T s";
WaterHH11:InputStep = .01;
WaterHH11:KpX = .093;
WaterHH11:T = 2.29;
WaterHH11:Kp = 9.3;
WaterHH11:b0 = 9.3;
WaterHH11:a1 = 2.29;
WaterHH11:Yo = -0.51838;

    /*************************************
     **** INSTANCE: WaterHH12
     *************************************/
MakeInstance( WaterHH12, PT1 );
WaterHH12:OutputInstance = WaterH;
WaterHH12:OperatingPointIn = .1621;
SetSlotOption( WaterHH12:OperatingPointIn, IMAGE, Edit10 );
WaterHH12:OperatingPointOut = 1.094;
SetSlotOption( WaterHH12:OperatingPointOut, IMAGE, Edit11 );
WaterHH12:DeadTimeInfo = N;
WaterHH12:Td = 0;
WaterHH12:ComplexPoles = No;
WaterHH12:Numerator = Kp;
WaterHH12:Denominator = "1 + T s";
WaterHH12:InputStep = .01;
```

```
SetSlotOption( WaterHH12:InputStep, IMAGE, Edit17 );
WaterHH12:KpX = .131;
SetSlotOption( WaterHH12:KpX, IMAGE, Edit12 );
WaterHH12:T = 7.08;
SetSlotOption( WaterHH12:T, IMAGE, Edit13 );
WaterHH12:Kp = 13.1;
WaterHH12:b0 = 13.1;
WaterHH12:a1 = 7.08;
WaterHH12:Yo = -1.02951;

    /*************************************
     **** INSTANCE: WaterHH1
     *************************************/
MakeInstance( WaterHH1, PT1 );
WaterHH1:OutputInstance = WaterH;
WaterHH1:OperatingPointOut = .131;
SetSlotOption( WaterHH1:OperatingPointOut, IMAGE, Edit11 );
WaterHH1:OperatingPointIn = .04;
SetSlotOption( WaterHH1:OperatingPointIn, IMAGE, Edit10 );
WaterHH1:DeadTimeInfo = N;
WaterHH1:Td = 0;
WaterHH1:ComplexPoles = No;
WaterHH1:Numerator = Kp;
WaterHH1:Denominator = "1 + T s";
WaterHH1:InputStep = .01;
WaterHH1:KpX = .057;
WaterHH1:T = .39;
WaterHH1:Kp = 5.7;
WaterHH1:b0 = 5.7;
WaterHH1:a1 = .39;
WaterHH1:Yo = -0.097;
WaterHH1:UndefinedParam = No;
SetSlotOption( WaterHH1:tq, IMAGE, Edit15 );
SetSlotOption( WaterHH1:Yq, IMAGE, Edit16 );

    /*************************************
     **** INSTANCE: WaterHH4
     *************************************/
MakeInstance( WaterHH4, PT1 );
WaterHH4:OutputInstance = WaterH;
WaterHH4:OperatingPointIn = .1158;
SetSlotOption( WaterHH4:OperatingPointIn, IMAGE, Edit10 );
WaterHH4:OperatingPointOut = 1.094;
SetSlotOption( WaterHH4:OperatingPointOut, IMAGE, Edit11 );
WaterHH4:DeadTimeInfo = N;
WaterHH4:Td = 0;
WaterHH4:ComplexPoles = No;
WaterHH4:Numerator = Kp;
WaterHH4:Denominator = "1 + T s";
WaterHH4:InputStep = .01;
SetSlotOption( WaterHH4:InputStep, IMAGE, Edit17 );
WaterHH4:KpX = .181;
SetSlotOption( WaterHH4:KpX, IMAGE, Edit12 );
WaterHH4:T = 8.47;
SetSlotOption( WaterHH4:T, IMAGE, Edit13 );
```

```
WaterHH4:Kp = 18.1;
WaterHH4:b0 = 18.1;
WaterHH4:a1 = 8.47;
WaterHH4:Yo = -1.00198;
WaterHH4:UndefinedParam = No;

    /*************************************
     **** INSTANCE: WaterHH15
     *************************************/
MakeInstance( WaterHH15, LinAttrib );
WaterHH15:SteadyState? = Yes;
WaterHH15:Overshoot? = No;
WaterHH15:Oscillation? = No;
WaterHH15:NoLag? = No;
WaterHH15:IncreasingResponseSpeed? = No;
WaterHH15:DecreasingResponseSpeed? = Yes;
WaterHH15:NonMinPhase? = No;

    /*************************************
     **** INSTANCE: WaterHH14
     *************************************/
MakeInstance( WaterHH14, NLdAttrib );
WaterHH14:LimitRateOfChange? = No;
WaterHH14:DeadTime? = No;
WaterHH14:VariantDeadTime? = No;
WaterHH14:VariantLimitRateChange? = No;
WaterHH14:VariantTimeConst? = Yes;
WaterHH14:DynamicNL? = Yes;

    /*************************************
     **** INSTANCE: WaterHH13
     *************************************/
MakeInstance( WaterHH13, NLsAttrib );
WaterHH13:Wiener? = Yes;
WaterHH13:Hammerstein? = No;
WaterHH13:StaticNL? = Yes;
WaterHH13:MultiValued? = No;
WaterHH13:MultiVariable? = Yes;
WaterHH13:TimeVariant? = No;
WaterHH13:Discontinuous? = No;
WaterHH13:SlopePos? = Yes;
WaterHH13:SlopeNeg? = No;
SetValue( WaterHH13:SlopePosDataU, .05, .1, .15 );
SetValue( WaterHH13:SlopePosDataY, .23, .83, 2. );
WaterHH13:SaturationUpper? = No;
WaterHH13:SaturationLower? = No;
WaterHH13:Minima? = Yes;
WaterHH13:DeadZones? = No;
WaterHH13:Maxima? = No;
WaterHH13:Output = WaterH;
WaterHH13:OutputInstance = WaterH;
SetValue( WaterHH13:MinimaDataU, 0.025 );
WaterHH13:MinimaNo = 1;
SetValue( WaterHH13:MinimaDataY, 0 );
```

```
    /*************************************
     **** INSTANCE: WaterHH19
     *************************************/
MakeInstance( WaterHH19, NLsAttrib );
WaterHH19:Wiener? = Yes;
WaterHH19:Hammerstein? = No;
WaterHH19:StaticNL? = Yes;
WaterHH19:MultiValued? = No;
WaterHH19:MultiVariable? = Yes;
WaterHH19:TimeVariant? = No;
WaterHH19:Discontinuous? = No;
WaterHH19:SlopePos? = Yes;
WaterHH19:SlopeNeg? = No;
SetValue( WaterHH19:SlopePosDataU, .05, .1, .15 );
SetValue( WaterHH19:SlopePosDataY, .1, .4, .95 );
WaterHH19:SaturationUpper? = No;
WaterHH19:SaturationLower? = No;
WaterHH19:Minima? = Yes;
WaterHH19:DeadZones? = No;
WaterHH19:Maxima? = No;
WaterHH19:Output = WaterH;
WaterHH19:OutputInstance = WaterH;
SetValue( WaterHH19:MinimaDataU, 0.03 );
WaterHH19:MinimaNo = 1;
SetValue( WaterHH19:MinimaDataY, 0 );

    /*************************************
     **** INSTANCE: WaterHH20
     *************************************/
MakeInstance( WaterHH20, NLsAttrib );
WaterHH20:Wiener? = Yes;
WaterHH20:Hammerstein? = No;
WaterHH20:StaticNL? = Yes;
WaterHH20:MultiValued? = No;
WaterHH20:MultiVariable? = Yes;
WaterHH20:TimeVariant? = No;
WaterHH20:Discontinuous? = No;
WaterHH20:SlopePos? = Yes;
WaterHH20:SlopeNeg? = No;
SetValue( WaterHH20:SlopePosDataU, .05, .1, .15 );
SetValue( WaterHH20:SlopePosDataY, .08, .23, .45 );
WaterHH20:SaturationUpper? = No;
WaterHH20:SaturationLower? = No;
WaterHH20:Minima? = Yes;
WaterHH20:DeadZones? = No;
WaterHH20:Maxima? = No;
WaterHH20:Output = WaterH;
WaterHH20:OutputInstance = WaterH;
SetValue( WaterHH20:MinimaDataU, 0.04 );
WaterHH20:MinimaNo = 1;
SetValue( WaterHH20:MinimaDataY, 0 );
```

List of References

[1] M. Rimvall, "Interactive environments for CACSD software", *4th IFAC Symp. on Computer Aided Design of Control Systems*, Beijing, pp. 17-26, 1988

[2] N. Munro, C. P. Jobling, "Ecstasy: A control system CAD environment", In D. A. Linkens (ed.), *CAD for Control Systems*, Marcel Dekker, New York, pp. 449- 467, 1993

[3] P. W. Grant, C. P. Jobling, C. Rezvani, "PRODYNSA - a package of tools for linear dynamic systems analysis written in PROLOG", *Control 91 - International Conference Publication*, The Institution of Electrical Engineers, vol. 1, no. 332, pp. 294-299, 1991

[4] H. A. Barker, M. Chen, P. W. Grant, I. T. Harvey, C. P. Jobling, P. Townsend, "The impact of recent developments in user-interface specification, design and management on the computer-aided design of control systems", *11th Triennial World Congress*, Tallinn, Estonia, IFAC - International Federation of Automatic Control, vol. 5, pp. 393-398, 1990

[5] H. A. Barker, M. Chen, P. W. Grant, C. P. Jobling, D. A. Simon, P. Townsend, "The manipulation of graphical and symbolic models of dynamic systems", *Analysis, design and evaluation of man-machine systems*, IFAC - International Federation of Automatic Control, no. 3, pp. 301-305, 1989

[6] H. A. Barker, M. Chen, P. W. Grant, C. P. Jobling, P. Townsend, "A man-machine interface for computer-aided design and simulation of control systems", *Automatica*, Pergamon Press, vol. 25, no. 2, pp. 311-316, 1989

[7] M. Chen, *Graphical man-machine interface to CACSD*, PhD Thesis University of Wales, Swansea, 1991

[8] H. A. Barker, M. Chen, P. W. Grant, C. P. Jobling, P. Townsend, "Graphical user interfaces in computer-aided control system design", In D. A. Linkens (ed.), *CAD for Control Systems*, Marcel Dekker, New York, pp. 561-577, 1993

[9] R. A. King, J. O. Gray, "A methodology for the design and implementation of graphical man-machine interfaces", *Knowledge engineering and computer modelling in CAD*, Butterworths, pp. 419-425, 1986

[10] P. H. M. Li, J. O. Gray, "Graphical system input for design", In D. A. Linkens (ed.), *CAD for Control Systems*, Marcel Dekker, New York, pp. 531-559, 1993

[11] H. Elmqvist, S. E. Mattsson, "Simulator for dynamical systems using graphics and equations for modeling", *IEEE Control Systems Magazine*, pp. 53-58, Jan. 1989

[12] F. E. Cellier, H. Elmqvist, "Automated formula manipulation supports object-oriented continuous-system modeling", *IEEE Control Systems Magazine*, vol. 13, no. 2, pp. 28-38, Apr. 1993

[13] H. Elmqvist, *Dymola - Dynamic Modeling Language - User's Manual*, Dynasim AB, Research Park Ideon, S-223 70 Lund, Sweden, 1994

[14] M. Otter, "DSblock specification", DLR, German Aerospace Research Establishment, Postfach 1116, 82230 Wessling, Germany

[15] G. Gruebel, C. P. Jobling, "Simulation tool-abstractions based on the neutral dynamics-model description format DSblock", *IEE Int. Conf. Control '94*, Conf. Publ. No. 389, pp. 1-6, 1994

[16] H. A. Barker, et al., "The making of EXCES - a software engineering perspective", *IFAC Computer Aided Design in Control Systems*, Swansea, UK, pp. 47-52, 1991

[17] P. J. Gawthrop, "Symbolic modeling in control", In D. A. Linkens (ed.), *CAD for Control Systems*, Marcel Dekker, New York, pp. 127-146, 1993

[18] M. Andersson, "An object-oriented language for model representation", *CACSD '89*, Tampa USA, pp. 8-15, Dec. 1989

[19] M. Andersson, "Discrete event modelling and simulation in Omola", *Proc. of the IEEE Symp. on Comp.-Aided Control Syst. Design*, pp. 262-268, 1992

[20] M. Andersson, *Omola - an object-oriented language for model representation*, Licentiate Thesis, Lund Institute of Technology, 1990

[21] J. H. Taylor, M. Rimvall, H. A. Sutherland, "Computer-aided control engineering environments", In D. A. Linkens (ed.), *CAD for Control Systems*, Marcel Dekker, New York, pp. 423-447, 1993

[22] C. P. Jobling, "Building better graphical user interfaces for CACSD - the case for object-oriented programming", *IFAC Computer Aided Design in Control Systems*, pp. 147-151, 1991

[23] C. P. Jobling, P. W. Grant, H. A. Barker, P. Townsend, "Object-oriented programming in control system design: a survey", *Automatica*, Pergamon Press, vol. 30, no. 8, pp. 1221-1261, 1994

[24] H. A. Barker, P. W. Grant, I. T. Harvey, C. P. Jobling, P. Townsend, "An approach to project management in computer aided control engineering", *Control Eng. Practice*, Pergamon, Elsevier Science Ltd, vol. 4, no. 4, pp. 441-453, 1996

[25] G. Gruebel, "AnDeCS: a concurrent control engineering project", *Proc. of the IEEE Symp. on Comp.-Aided Control Syst. Design*, pp. 37-46, 1992

[26] G. Gruebel, "The ANDECS CACE framework", *IEEE Control Systems*, vol. 15, no. 2, pp. 8-13, 1995

[27] M. Rimvall, M. Kuendig, "Intelligent help for CACE applications", *11th Triennial World Congress*, Tallinn, Estonia, IFAC - International Federation of Automatic Control, vol. 5, pp. 367-372, 1990

[28] H. Hvelplund, "The ESPRIT Eurohelp project: an intelligent help system", *Proc. of the Int. Conf. on Knowledge Based Syst.*, pp. 61-70, 1986

[29] G. Hoffmann, M. Rimvall, "Knowledge representation in computer-aided control systems design packages", *Proc. of the Euro. Simulation Multi-Conf.*, pp. 223-227, 1988

[30] H. Meier zu Farwig, H. Unbehauen, "Knowledge-based system identification", *IFAC Symposium on Identification and System Parameter Estimation*, pp. 21-29, 1991

[31] J. H. Taylor, "Expert-aided environments for CAE of control systems", *IFAC Conf. on Comp. Aided Design in Control Systems*, pp. 7-16, 1988

[32] X. Li, C. P. Jobling, P. W. Grant, "Intelligent object-oriented modelling", *IEE Colloq. on Advanced CACSD*, Digest No. 96/061, pp. 6/1-6/3, 1996

[33] X. Li, C. P. Jobling, P. W. Grant, "An object-oriented information model for intelligent modelling", *IFAC'96 World Congress*, San Francisco, pp. 483-488, 1996

[34] J. Rumbaugh, et al., *Object-oriented modeling and design*, Prentice-Hall, Englewood Cliffs, NJ, 1991

[35] M. T. Rahbar, S. Bennet, D. A. Linkens, "Strategies for designing a knowledge based environment for modelling and simulation", *UKSC'90*, Brighton, pp. 146-149, 1990

[36] D. A. Linkens, et al., "Experiences from a prototype environment for intelligent modelling and simulation", *IFAC Computer Aided Design in Control Systems*, Swansea, UK, pp. 59-64, 1991

[37] E. B. Tanyi, D. A. Linkens, S. Bennett, "The use of Prolog in the implementation of a knowledge-based environment for modelling and simulation (KEMS)", *Trans. Inst. Measurement Control*, vol. 15, no. 5, pp. 248-259, 1993

[38] E. B. Tanyi, D. A. Linkens, "Addition of a knowledge acquisition facility to a knowledge based environment for modelling and simulation (KEMS)", *Proc. of 3rd Europ. Sim. Congress*, UK Simulation Council, pp. 193-197, 1989

[39] E. B. Tanyi, D. A. Linkens, S. Bennett, "The use of a knowledge acquisition module (KAM) for constructing simulation models", *UKSC'90*, Brighton, pp. 150-153, 1990

[40] E. B. Tanyi, D. A. Linkens, S. Bennett, "Knowledge acquisition and hierarchical structures for modelling and simulation", *Artificial Intelligence, Expert Systems and Symbolic Computing*, Elsevier Science Publishers B V, North-Holland, pp. 60-69, 1992

[41] M. R. Smith, D. A. Linkens, S. Bennett, "Model validation for a knowledge-based environment for modelling and simulation (KEMS)", *UK Simulation Council '90 Conference*, Brighton, pp. 50-53, 1990

[42] W. M. Austin, B. Khoshnevis, "Qualitative modeling using natural language: An application in system dynamics", *Qualitative Simulation Modeling and Analysis* (eds.: P A Fishwick, P A Luker), Springer-Verlag, pp. 263-301, 1991

[43] A. Niederlinski, J. Kasprzyk, J. Figwer, "Intelligent identification: decision trees or knowledge bases?", *IFAC Symposium on System Identification*, 1994

[44] B. Bieker, *Beitraege zu Wissenserwerb und Wissensrepraesentation fuer Experten-Regler*, Doctorate Dissertation, University of Munich, Germany, 1988

[45] F. Puppe, *Systematic introduction to expert systems*, Springer-Verlag, 1993

[46] G. Nowinski, *Wissensbasierte Regelungen mit effizienten Wissenserwerbsmethoden*, Doctorate Dissertation, University of Karlsruhe, Germany, VDI Fortschrittberichte, 1993

[47] M. Krabs, *Das ROSA-Verfahren zur Modellierung dynamischer Systeme durch Regeln mit statistischer Relevanzbewertung*, Doctorate Dissertation, University of Dortmund, Germany, Fortschrittberichte VDI, 1994

[48] A. Scott, D. A. Linkens, S. Bennett, M. Smith, "The use of expert-ease's "EASE+" package to develop user-interfaces for mathematical modelling and simulation", *IEE Colloq Software Tools for Interface Design*, pp. 14/1-14/3, 1990

[49] D. A. Linkens, "An artificial intelligence approach to environments for modelling and simulation", In D. A. Linkens (ed.), *CAD for Control Systems*, Marcel Dekker, New York, pp. 71-105, 1993

[50] L. Bainbridge, "Verbal reports as evidence of the process operator's knowledge", *Fuzzy Reasoning and its Applications*, Academic Press, pp. 343-368, 1981

[51] R. Isermann, *Identifikation dynamischer Systeme 1+2*, Springer-Verlag, 1992

[52] P. Eykhoff, *System identification*, John Wiley & Sons, 1974

[53] R. P. Lippmann, "An introduction to computing with neural nets", *IEEE ASSP Magazine*, pp. 4-22, 1987

[54] Y. Sawaragi, Y. Nakamori, "Computer aided modeling and simulation in process control systems", *IFAC Computer Aided Design in Control Systems*, Swansea, UK, pp. 227-232, 1991

[55] Y. Nakamori, "Development and application of an interactive modeling support system", *Automatica*, Pergamon Press, vol. 25, no. 2, pp. 185-206, 1989

[56] J. Lunze, "Bemerkungen zur qualitativen Beschreibung dynamischer Systeme", *Automatisierungstechnik - at*, R. Oldenbourg Verlag, vol. 40, no. 8, pp. 281-284, 1992

[57] J. Lunze, "Ein Ansatz zur qualitativen Modellierung und Regelung dynamischer Systeme", *Automatisierungstechnik - at*, R. Oldenbourg Verlag, vol. 41, no. 12, pp. 451-460, 1993

[58] J. De Kleer and J. S. Brown, "A qualitative physics based on confluences", *Artificial Intelligence*, North-Holland Publishing Company, vol. 24, pp. 7-83, Dec., 1984

[59] K. D. Forbus, "Qualitative process theory", *Artificial Intelligence*, North-Holland Publishing Company, vol. 24, pp. 85-167, Dec., 1984

[60] P. A. Fishwick, "Fuzzy simulation: specifying and identifying qualitative models", *Int. J. General Systems*, vol. 19, pp. 295-316, 1991

[61] A. L. Stevens et al., "A program for handling multiple phases of the design cycle in process control system design", *Engng Applic. Artif. Intell.*, Pergamon Press, vol. 6, no. 3, pp. 231-239, 1993

[62] M. L. Mavrovouniotis and G. Stephanopoulos, "Formal order-of-magnitude reasoning in process engineering", *Comput. Chem. Engng.*, vol. 12, no. 9/10, pp. 867-880., 1988

[63] M. L. Mavrovouniotis, "A belief framework for order-of-magnitude reasoning", submitted for publication, 1994, please contact Prof. Dr. Mavrovouniotis at Northwestern University, Chem. Eng. Department, Evanston, IL 60208-3120.

[64] P. J. Hayes, "The naive physics manifesto", D. Michie (Ed.): *Expert systems in the microelectronic age*, Edinburgh University Press, 1978

[65] B. C. Williams, "MINIMA a symbolic approach to qualitative algebraic reasoning", *Proc. AAAI 88*, pp. 264-269, 1988

[66] R. Simmons, "'Commonsense' arithmetic reasoning", *Proc. AAAI 86*, pp. 118-124, 1986

[67] K. D. Forbus, "The qualitative process engine", in Weld and De Kleer (Eds.): *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufmann, pp. 220-235, 1990

[68] B. Falkenhainer and K. D. Forbus, "Compositional modelling of physical systems", *Recent advances in qualitative physics*, MIT Press, pp. 33-48, 1992

[69] K. D. Forbus and B. Falkenhainer, "Self-explanatory simulations: integrating qualitative and quantitative knowledge", *Recent advances in qualitative physics*, MIT Press, pp. 49-65, 1992

[70] B. D'Ambrosio, *Qualitative process theory using linguistic variables*, PhD-Thesis, University of California, Berkeley, 1986

[71] B. D'Ambrosio, *Qualitative process theory using linguistic variables: symbolic computing*, Springer-Verlag, 1989

[72] B. Kuipers, "Commonsense reasoning about causality: deriving behaviour from structure", *Artificial Intelligence*, North-Holland Publishing Company, vol. 24, pp. 169-203, Dec., 1984

[73] B. Kuipers, "Qualitative simulation as causal explanation", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-17, no. 3, 1987

[74] D. T. Dalle Molle, B. J. Kuipers and T. F. Edgar , "Qualitative modelling and simulation of dynamic systems", *Comput. Chem. Engng.*, Pergamon Press plc, vol. 12, no. 9/10, pp. 853-866, 1988

[75] B. Kuipers, "Qualitative simulation", *Artificial Intelligence*, Elsevier Science Publishers B.V. (North-Holland), vol. 29, pp. 289-338, 1986

[76] B. Kuipers, *Qualitative reasoning: modeling and simulation with incomplete knowledge*, MIT Press, 1994

[77] B. Kuipers and D. Berleant, "Using incomplete quantitative knowledge in qualitative reasoning", *Proc. AAAI-88*, pp. 324-329, 1988

[78] D. Berleant and B. Kuipers, "Qualitative-numeric simulation with Q3", *Recent advances in qualitative physics*, MIT Press, pp. 3-16, 1992

[79] H. Kay and B. Kuipers, "Numerical behavior envelopes for qualitative models", *Proc. of the 11th Nat. Conf. on Artificial Intelligence (AAAI-93)*, MIT Press, pp. 606-613, 1993

[80] A. Farquhar, *Automated modelling of physical systems in the presence of incomplete knowledge*, PhD Thesis, University of Texas at Austin, 1993

[81] J. Crawford, A. Farquhar and B. Kuipers, "QPC: A compiler from physical models into qualitative differential equations", *Recent advances in qualitative physics*, MIT Press, pp. 17-32, 1992

[82] A. Farquhar and G. Brajnik, "A semi-quantitative physics compiler", *Working Papers of the Int. Workshop on Qualitative Reasoning (QR-94)*, 1994

[83] O. Raiman, "Order of magnitude reasoning", *Proc. AAAI*, pp. 100-104, 1986

[84] Q. Shen and R. Leitch, "Fuzzy qualitative simulation", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 4, pp. 1038-1061, 1993

[85] Q. Shen and R. Leitch, "Combining qualitative simulation and fuzzy sets", *Recent advances in qualitative physics*, MIT Press, pp. 83-100, 1992

[86] G. M. Coghill and M. J. Chantler, "Mycroft: a framework for qualitative reasoning", *Intelligent Systems Engineering*, IEE Conference Publ. No. 395, pp. 43-48, Sept. 1994

[87] L. A. Zadeh, "Fuzzy sets", *Info and Control*, no. 8, pp. 338-353, 1965

[88] J. E. Hopcroft and J.D. Ullmann, *Introduction to automata theory, languages and computation*, Addison-Wesley Publishing Company, 1979

[89] A. Bredebusch, J. Lunze and H. Richter, "A Petri-Net representation of the qualitative behaviour of a dynamical continuous-time system", *Intelligent Systems Engineering*, IEE Conference Publ. No. 395, pp. 223-228, Sept. 1994

[90] Lunze, J, "A Petri-net approach to qualitative modelling of continuous dynamical systems", *SAMS*, Gordon and Breach Science Publishers, vol. 9, pp. 89-111, 1992

[91] M. Kluwe, V. Krebs, J. Lunze and H. Richter, "Beratungssystem fuer die operative Prozessfuehrung", *Automatisierungstechnische Praxis atp*, vol. 36, no. 8, pp. 11-16, 1994

[92] V. Krebs, M. Kluwe, J. Lunze and H. Richter, "Ein hybrides Modell zur qualitativen und quantitativen Beschreibung komplexer dynamischer Systeme", *Berichte vom 39. Internationalen Wissenschaftlichen Kolloquium in Ilmenau*, Verlag: Universitaetsbibliothek der TU Ilmenau, vol. 3, pp. 159-166, 1994

[93] M. Kluwe, V. Krebs, J. Lunze and H. Richter, "Qualitative modelling based on rules, Petri-nets, and differential equations", *Proc. on IMACS Symposium on Mathematical Modelling 1. Mathmod*, Vienna, Technical University Vienna, vol. 1, pp. 134-137, 1994

[94] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling", *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, Feb., 1993

[95] D. T. Dalle Molle, *Qualitative simulation of dynamic chemical processes*, PhD-Thesis, University at Austin, TX, 1989

[96] B. C. Williams, "Doing time: putting qualitative reasoning on firmer ground", *AAAI-86 Conference*, Philadelphia PA, pp. 105-112, 1986

[97] H. Ushida, T. Yamaguchi, K. Goto and T. Takagi, "Fuzzy-neuro control using associative memories, and its applications", *Control Eng. Practice*, Pergamon Press Ltd, vol. 2, no. 1, pp. 129-145, 1994

[98] K. Goto, T. Yamaguchi and T. Takagi, "Dynamic model for a plant using associative memory system", *Artificial Neural Networks*, 2 - Proc. of the '92 Int. Conf. (ICANN-92), Elsevier Science Publishers B.V., pp. 1517-1520, 1992

[99] M. Sugeno and G. T. Kang, "Fuzzy modelling and control of multilayer incinerator", *Fuzzy Sets and Systems*, Elsevier Publishers B.V., vol. 18, pp. 329-345, 1986

[100] R. Rajagopalan, "Qualitative modelling and simulation: A survey", *AI applied to Simulation*, Proc. of Europ. Conf., pp. 9-26, 1986

[101] D. A. Linkens, "AI in control systems engineering", *The Knowledge Engineering Review*, no. 5:3, pp. 181-214, 1990

[102] R. Leitch, "Qualitative modeling in control", In D. A. Linkens (ed.): *CAD for Control Systems*, Marcel Dekker, New York, pp. 107-125, 1993

[103] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control", *IEEE Trans. on Syst., Man, and Cybernetics*, vol. SMC-15, no. 1, pp. 116-132, 1985.

[104] OMRON Corporation (Japan), *Fuzzy Guide Book*, 1992.

[105] H.-J. Zimmermann, *Fuzzy set theory - and its applications*, Kluwer Academic Publishers, 1991

[106] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes", IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-3, no. 1, pp. 28-44, 1973.

[107] M. Strickrodt, R. Schumann and K. Baker, "An integrated knowledge engineering approach to process modelling for CACSD", *Proceedings of IFAC'96 World Congress*, 1996.

[108] M. Strickrodt, R. Schumann, K. Baker, " Knowledge acquisition as part of a practical CACSD approach", *IEE Colloq. on Advances in CACSD*, Digest No: 96/061, pp. 8/1-8/4, 1996.

[109] A. Hammerstein, "Nichtlineare Integralgleichungen nebst Anwendungen", (in German), *Acta Mathematica*, vol. 54, pp. 117-176, 1930.

[110] S. A. Billings, S. Y. Fakhouri, "Identification of systems composed of linear dynamic and static nonlinear elements", *5th IFAC Symp. on Ident. and Syst. Par. Est.*, pp. 493-500, 1979.

[111] R. Jain, "Analysis of fuzzy systems", *Fuzzy Automata and Decision Process*, pp. 251-268, 1977.

[112] G. B. Grobbelaar, "The use of fuzzy sets theory for modelling complex processes", *The Transactions of the SA Institute of Electrical Engineers*, vol. 81, no. 2, pp. 18-21, Jun., 1990.

[113] A. Kandel, "Imprecise modelling of systems and circuits in uncertain environments", *Proc. of the 1981 Europ. Conf. on Circuit Theory and Design*, Delft Univ. Press, pp. 984-987, 1981.

[114] R. Rouhani, "Bounding the behaviour of positive dynamic systems", *Journal of Economic Dynamics and Control*, vol. 2, pp. 283-299, 1980.

[115] D. Dubois and H. Prade, *Fuzzy sets and systems: theory and applications*, Academic Press, N.Y., 1980, chapter 2.

[116] B. Kuipers and K. Åström, "The composition and validation of heterogeneous control laws", *Automatica*, Pergamon, Elsevier Ltd., vol. 30, pp. 233-249, 1994.

[117] B.-M. Pfeiffer, "Selbsteinstellende klassische Regler mit Fuzzy-Logik", (in German), *Automatisierungstechnik*, vol. 42, no. 2, pp. 69-73, 1994.

[118] D. J. Murray-Smith, "A review of methods for the validation of continuous system simulation models", *UKSC'90*, Brighton, pp. 108-111, 1990

[119] Booch, *Object oriented design with applications*, The Benjamin/Cummings Publishing Company Inc., 1991

[120] M. A. Jackson, *Principles of program design*, Academic Press Inc., 1975

[121] T. De Marco, *Structured analysis and system specification*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979

[122] D. De Champeaux, P. Faure, "A comparative study of object-oriented analysis methods", *Journal of Object-Oriented Programming*, vol. 5, pp. 21-32, 1992

[123] The SCS Technical Committee on Model Credibility, "Terminology for model credibility", *Simulation*, vol. 32, no. 3, pp. 103-104, 1979

[124] KAPPA-PC manual, Intellicorp, 1994

[125] R. Schumann, "CAE von Regelsystemen mit IBM-kompatiblen Personal Computern" ("CAE of control systems with IBM-compatible personal computers"), *Automatisierungstechnische Praxis,* Oldenbourg Verlag, vol. 31, no. 8, pp. 349-359, 1989

[126] R. Schumann, "CAE von Regelsystemen mit IBM-kompatiblen Personal Computern", *Automatisierungstechnische Praxis,* Oldenbourg Verlag, vol. 33, no. 3, pp. 147-152, 1991

[127] R. Schumann, "CAE von Regelsystemen mit IBM-kompatiblen Personal Computern", *Automatisierungstechnische Praxis,* Oldenbourg Verlag, vol. 34, no. 8, pp. 467-471, 1992

[128] C. P. Jobling, "Advances in computer aided control system design", *IEE Colloq. on Advanced CACSD*, Digest No. 96/061, pp. 1/1-1/6, 1996

[129] G. Schwarze, *Regelungstechnik fuer Praktiker Formeln-Kurven-Tabellen,* Reihe Automatisierungstechnik, Vieweg & Sohn, Braunschweig, vol. 50, 1968

[130] T. Varsamidis, S. Hope, C. P. Jobling, "Integration using a unified model for CACSD", *IEE Colloq. on Advanced CACSD,* 1996, Digest No. 96/061, pp. 2/1-2/4

[131] H. A. Barker, M. Chen, P. W. Grant, C. P. Jobling, P. Townsend, "Modern environments for dynamic system modelling", *Int. Journal of Modelling & Simulation,* 1993, vol. 13, no. 2, pp. 67-71

Manuscripts Submitted For Publication

The below listed manuscripts are appended after this page for reference.

- Strickrodt, M., R. Schumann and K. Baker, "An integrated knowledge engineering approach to process modelling for CACSD", *Proceedings of IFAC'96 World Congress*, 1996.

- Schumann, R., S. Körner, K. Baker, M. Strickrodt, "Shaping CACSD for practical use in process industry", *Proceedings of IFAC'96 World Congress*, 1996.

- Strickrodt, M., R. Schumann, K. Baker, " Knowledge acquisition as part of a practical CACSD approach", *IEE Colloq. on Advances in CACSD*, Digest No: 96/061, pp. 8/1-8/4, 1996.

- Strickrodt, M., K. Baker, "A fuzzy hybrid model for the simulation of nonlinear dynamic processes", accepted for publication in the *IEEE Transactions on Systems, Man and Cybernetics*.

- Strickrodt, M., K. Baker, "Practical applicability of qualitative modelling approaches to process simulation: A survey", submitted for publication in the *IEEE Transactions on Systems, Man and Cybernetics*.

AN INTEGRATED KNOWLEDGE ENGINEERING APPROACH TO PROCESS MODELLING FOR CACSD

M. Strickrodt*, R. Schumann and K. J. Baker***

* *University of Glamorgan, Department of Mech. & Man. Engineering, Pontypridd, Mid Glamorgan CF37 1DL, U.K., Tel. +44-(0)1443-480480, e-mail: mstrickr@glam.ac.uk*
** *Fachhochschule Hannover, Fachgebiet Automatisierungstechnik, REPAM, Ricklinger Stadtweg 120, 30459 Hannover, Germany, Tel. +49-(0)511-9296-311*

Abstract: An integrated approach to process modelling on the basis of knowledge and experience from process engineers is suggested. This approach is designed in such a way that its implementation into an interactive computer program can be applied by process experts who usually have little or no experience in process modelling. 'Integrated' refers firstly to the idea to combine different types of process models into a single process modelling approach and secondly to the envisaged openness of the implementation towards existing computer aided control system design (CACSD) environments with respect to data and information exchange.

Keywords: computer-aided control system design, fuzzy hybrid systems, industrial control, knowledge engineering, modelling, process models

1. INTRODUCTION

1.1. The Need for a new Approach to Process Modelling

In general, there is not yet an approach existing which allows engineers - except for control or modelling experts - to build a process model. However, there is a need for a modelling approach which makes use of the practitioner's (i.e., area engineer's) experience in industry:

A) In small or medium size companies, where control experts are usually not available (apart from high technology industry), such a modelling approach is needed for the introduction of systematic approaches to control system design, simulation and optimisation which can be handled by the area engineers.
B) In bigger companies, this modelling approach is needed as a means of communicating the area engineer's process knowledge to the control experts.

The former aspect addresses the absence of systematic control system design and optimisation approaches in big parts of manufacturing industry where intuition and rule of thumb tuning still prevail. Despite the trend of providing more and more user friendly interfaces for the established Computer Aided Control System Design (CACSD) environments, these are still not geared at the skill level of area engineers in industry but mainly aim at an improved handling for control experts. This is not to say that all CACSD programs should be simplified to a level that inexperienced users could handle them but that

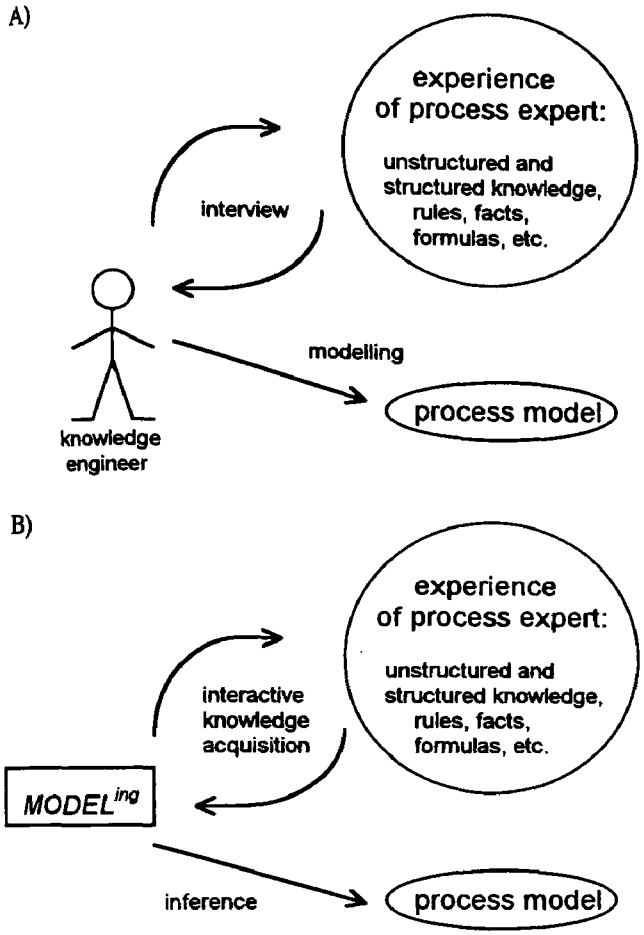


Fig. 1. The Knowledge Engineering Approach. Knowledge engineers (A) are normally not available in industry. The interactive *MODEL*^{ing} program (B) should eventually do this job.

there is still scope for solutions that bridge the gap between such users and advanced technology. The resulting widening gap between advanced control engineering methods and the application in many industries is of major economic importance.

The aspect B) on the other hand addresses the activation of a potential source of valuable practical information in companies with separate control engineering departments: so far, the control engineers in such corporate environments make only little use of the experience collected by the area engineers. The reason for this lies largely in the difficulties related to the translation of the practical experience into control relevant information. However, even the use of partially defined models which are derived from practical experience could significantly reduce costs and give new process insights, since theoretical process modelling is normally too time consuming and expensive for industrial applications. Although the practical experience of the area engineer should ideally be translated into a simplified but fully parameterised process model, the possibly only partially defined model could further be used as the important 'a priori' knowledge for improved process identification experiments.

Library based approaches have been developed to address this need (Linkens, et al., 1991; Rahbar, et al., 1990). Although this is a very sensible approach, it can only form part of the solution since any library is limited to pre-specified elements and it takes modelling experts to add new components. Furthermore, libraries are normally quite inflexible with respect to using different kinds of knowledge. Extensions of this work described by Tanyi et al. (1992) aimed at overcoming these deficiencies.

1.2. The overall Aim of the presented Research Project

The main objective is the development of a new, integrated approach to process modelling on the basis of knowledge and experience from area engineers which is referred to in the following as '*MODEL*^{ing}'. This project forms one part in a collaborative research project with the Fachhochschule in Hannover, Germany, with the overall aim of a simplified approach to Computer Aided Control System Design for industrial application and must therefore be seen in this context: other program modules will make use of the possibly partially defined process models which are obtained through the process-knowledge acquisition tool that is described in this paper.

The term 'knowledge engineering approach' as part of the title for this research project is descriptive in the sense that the aim is to largely replace the knowledge engineer who is, according to the understanding in artificial intelligence (AI), normally responsible for the translation of the unstructured area expert's knowledge into a formal representation (here: a process model), as Figure 1 illustrates.

Although the *MODEL*^{ing} approach will not be able to fully replace a knowledge engineer, it could overcome some of the weaknesses of the knowledge engineer's interview techniques by avoiding the need for the area engineer to formulate all experience in words.

Apart from the aim of making modelling techniques accessible for engineers with little experience in process modelling, both control experts and engineering students could benefit from this approach, too.

2. THE INTEGRATED MODELLING APPROACH

This chapter introduces the general concept of the suggested approach. As opposed to the introduction of either comprehensive conventional or 'intelligent' help functions (Hoffmann and Rimvall, 1988; Hvelplund, 1986), *MODEL*^{ing} aims at being largely self explanatory by making extensive use of graphical aids (Jobling, 1991).

2.1. The Modelling Sequence

Focussing primarily on the linear dynamics modelling part, the sequence is described here in some details since the novelty is not as much in *what* is featured in the *MODEL*^{ing} approach as in *how* the user's knowledge can be accessed.

The modelling of a new process component commences with the specification of a unique component name and its 'cuts' (i.e., input and output variables with their units) to maintain the consistency in the overall context of the process modelling.

The next step makes use of a graphical representation of step responses to access the process expert's understanding of the temporal behaviour of the process (Step-Responses Window, Fig. 2). Step responses are fairly straightforward to understand. By selecting the step response graph that matches the real process behaviour best, the area engineer gives indirectly a very detailed piece of information without being bothered with theoretical details. The set of implicit answers to be given by the selection of a step response, however, could exceed the abilities of the area engineer in some situations. The direct selection of an appropriate step response is therefore complemented with a decision support facility (Fig. 3).

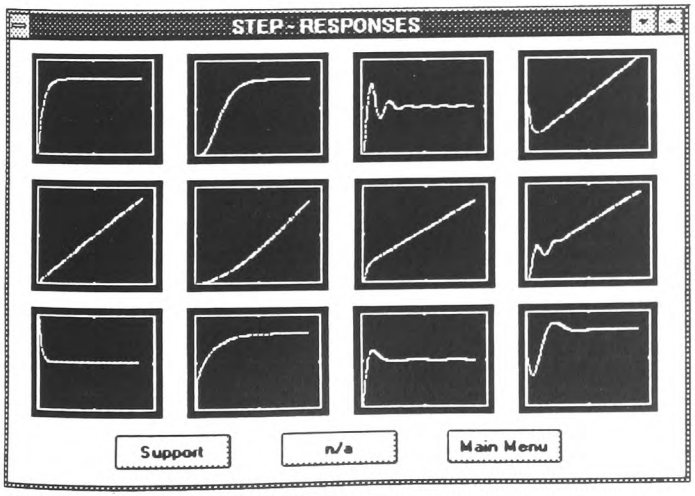


Fig. 2. Step Responses Window

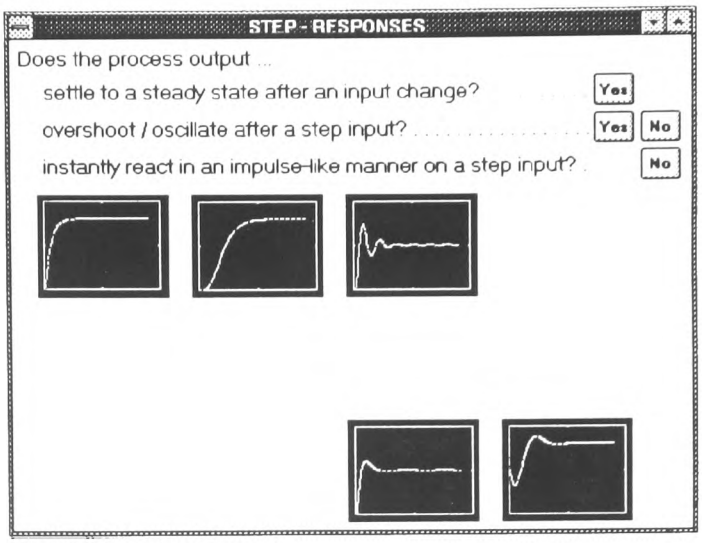


Fig. 3. Decision Support

With the decision support facility, the choice of possible step responses is incrementally narrowed. Its main characteristics are as follows:

- firstly, the three most important questions for the decision are explicitly asked
- the answers are either 'yes' or 'no' - in the *MODEL*^{ing} program, a 'yes' or 'no'-button is pressed; the selected option remains on the screen, the other disappears
- in the case that an answer is unknown, the question can remain unanswered
- every answer is instantly evaluated: inapplicable step responses disappear from the selection table - the direct implication of each piece of information on the decision process is watched by the user, which has a valuable educational effect
- the questions can be answered in an arbitrary order
- after all known answers are given, there is ideally only one step response left which is then selected; in the case that several step responses remain, a second set of questions appears
- the second set of questions depends on the answers given before so that only sensible and important questions for the specific situation are asked

At first sight, the decision support might seem to be rather trivial or especially geared at educational purposes. A closer look at systematic procedures for problem solving reveals, however, that it is most important to ask the right questions. This applies as much to problem analysis in everyday life as to troubleshooting in industry and to process modelling. Formulating the right questions in order to find the relevant answers becomes all the more difficult the less one knows about the problem domain. An experienced process engineer in industry, for example,

who has to analyse and solve a problem which occurred with one of the processes is perfectly able to formulate the problem-specific questions, find out the answers and thereby solve the problem. The same process engineer has probably sufficient process knowledge to answer several questions which are relevant to process modelling for control purposes, but without sufficient modelling knowledge he/she would not be able to ask the relevant questions in the first place.

Having applied the decision support a couple of times, the user will probably be able to make a direct decision in the future, which is an important educational aspect of the decision support. Also, it could well be that it is impossible to answer all the implicit questions of the direct selection in a given situation. In such a situation, the decision support enables the user to enter the partial knowledge which is available.

After the selection of an appropriate step response the user is requested to provide some numerical values of characteristic step response parameters which are illustrated in a separate graphical window (Fig. 4), so that a parameterised model can be calculated. The values to be entered can either be rough estimates or derived from process data, like quality assurance notes or a quick, simple test of the process. In the case when the requested parameter values can neither be estimated nor determined in some other way, qualitative defaults are set. These defaults can be used to calculate 'dummy' parameters for the transfer function so that the process can at least qualitatively be simulated using a numerical simulator. In this case, the numerical results of the simulation are meaningless - only the qualitative shape of the output trajectory is of importance. To simulate such numerically undefined process models in a truly qualitative manner, Mycroft (Coghill and Chantler, 1994) or QSIM (Kuipers,

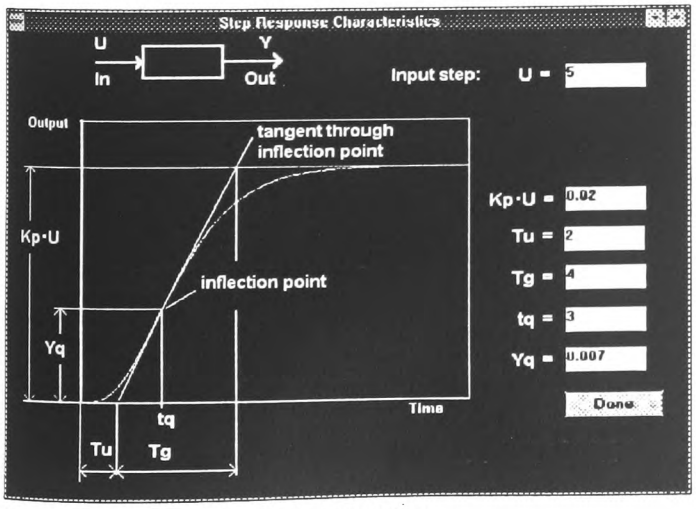


Fig. 4. Step Response Characteristics

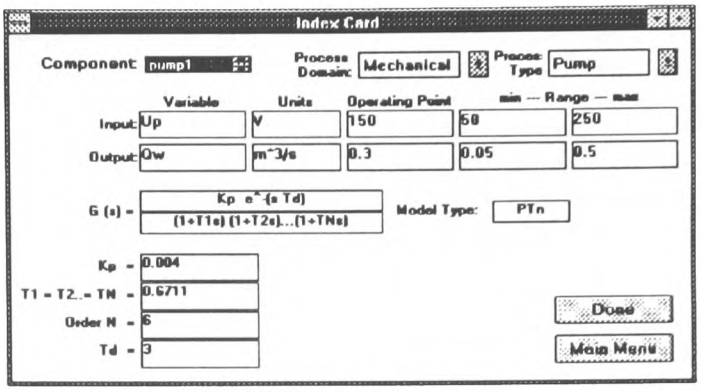


Fig. 5. 'Index Card' with Results

1987), with its later add-ons, like NSIM (Kay and Kuipers, 1993) might be applied at a later stage of this research project.

Finally, this linear *MODEL*^{ing} sequence returns to the 'Index Card' (Fig. 5) where the inference result, i.e., the process model is displayed in control engineering terms.

As it was mentioned above, this sequence for the modelling of linear dynamics is only one part of the knowledge acquisition approach. To allow for a simplified approach to the modelling of nonlinear processes in a similar fashion, a Fuzzy Hybrid modelling approach, which combines conventional transfer functions with the fuzzy notion, has been developed by the authors. This Fuzzy Hybrid concept is similar to the approach suggested by Takagi and Sugeno (1985), but particularly aimed at dynamic process modelling. It is therefore geared at the modelling of continuous processes whose static and/or dynamic characteristics vary, depending on one or more process parameters. 'Patchy' process knowledge - normally in the form of locally valid, piecewise linear equations, which are obtained through repetitive application of the above described linear modelling approach - are combined to a highly nonlinear global process model. Without any loss in generality, the approach, which is described in detail in a separate paper due to be published, can be largely combined with default settings with respect to fuzzy membership functions and implication method, so that the required modelling effort is minimal. Using this approach, processes with, for example, changing time constants, gains or damping ratios as well as multivariable relationships which are normally difficult to model are therefore easy to handle. Among the further advantages is the possibility to make use of the various traditional system analysis techniques and the modularity of the approach that allows for further improvements of the model by additional information at any later stage.

Ideally, the user is able to provide all the requested information for the nonlinear or linear modelling approach as described above. In many practical situations, however, the available knowledge will be rather more limited but nevertheless worthy to be exploited. The result of applying the *MODEL*^{ing} approach will therefore vary between:

- A fully parameterised approximate process model (linear or nonlinear), suitable for coarse numerical simulations and further refinement using identification techniques on the basis of process data.

  :

- A partially or qualitatively (e.g. in orders of magnitude) parameterised approximate process model, suitable for semi-quantitative or qualitative simulation and as a detailed basis for identification experiments.

  :

- A structurally defined approximate process model, suitable as a detailed basis for identification experiments.

  :

- Relationships between process parameters, formulated in a set of IF ... THEN ... rules (i.e., production rules).

  :

- A list of descriptive attributes that characterise the linear and nonlinear behaviour of the process; suitable as 'a priori' information for the design of optimal identification experiments.

In order to acquire the 'list of descriptive attributes', the above detailed linear/nonlinear *MODEL*^{ing} sequence is complemented by an interactive module that extracts the relevant attributes from the user's answers to a sequence of systematic and context related questions. Although this module is mainly aimed at the situation where only relatively little information is available, it is also useful for collecting supplementary information in the case that, for example, a partially parameterised approximate process model can be derived. Likewise, the rule-based modelling approach, which is another part of the sequence, can be taken on its own or as a means of supplementing information.

2.2. Simulation Code Generation

For the flexible further use of sufficiently defined process models which are derived from the process expert's knowledge using the *MODEL*^{ing} approach, a simulation code generator has been programmed. Using this function, simulation code in the form of an ASCII file is generated which enables the direct import of the process model into the block-oriented MATLAB™/Simulink™ simulation program. In the graphical block editor of Simulink, the process model is automatically shown as a component block with the name which was specified at the beginning of the *MODEL*^{ing} sequence (Fig. 6). This component block can either be simulated on its own to validate the behaviour of the process model or it can be integrated as one component of a more complex, structured model of a process.

In further pursuit of the 'open CACSD' philosophy, an interface to DYMOLA™, the object oriented simulation language with particular future potential, will be added. DYMOLA™ in turn provides further links to other environments, like ACSL™, as well as a translator to the neutral 'DSblock' specification (Otter, 1992).

3. CONCLUSION

An analysis of the current situation showed that for a more widespread application of CACSD systems in industry, tools that provide different levels of functionality for all potential users - from technicians with very basic control knowledge up to experts - are required. Within the currently existing CACSD programs, process modelling is an area of particularly poor support. New modelling tools for experts and practical users should therefore complement each other so that best use of the existing knowledge can be made.

As a contribution to improve this situation, the approach introduced in this paper aims at translating the process expert's knowledge about the static relationships and temporal behaviour of the process into technically useful forms. Depending on the individual modelling situation and the type of available information, one of the various modelling approaches - or even a combination of approaches - will be advantageous to the other ones. The seamless integration of the different modelling approaches into a single approach is the salient feature of *MODEL*^{ing}, which makes it interesting not only for area engineers but also for industrial control engineering departments and educational purposes.

Acknowledgements: The authors would like to thank Steffen Körner for the fruitful discussions. The research was carried out with the support of both the British Council and DAAD (Deutscher Akademischer Austauschdienst) through the Anglo-German Academic Research Collaboration Programme ARC, Project No. 520 and 313-ARC-VII-93/86/scu, respectively, for the collaboration between the University of Glamorgan and the Fachhochschule Hannover.

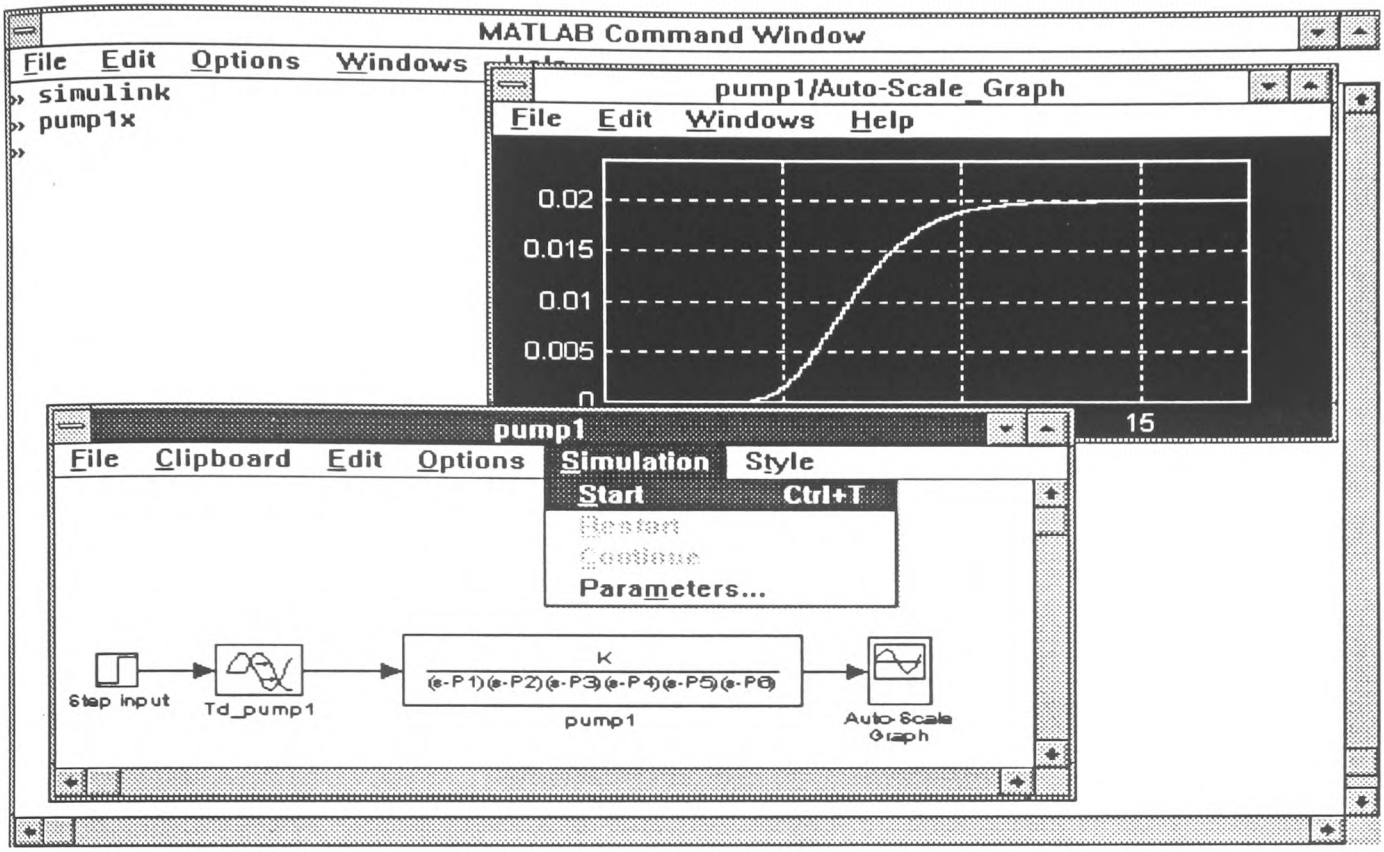


Fig. 6. Simulation of the obtained Process Model

REFERENCES

Coghill, G. M. and M. J. Chantler (1994). MYCROFT: a framework for qualitative reasoning. *Intell. Syst. Eng.*, IEE Conference Publ. No. 395, pp. 43-48.

Hoffmann, G. and M. Rimvall (1988). Knowledge representation in computer-aided control systems design packages. *Proc. of the Europ. Simul. Multi-Conf.*, SCS Eur., Ghent, Belgium, pp. 223-227.

Hvelplund, H (1986). The ESPRIT Eurohelp project: an intelligent help system. *Proc. of the Int. Conf. on Knowledge Based Syst.*, Online Publications, Pinner, UK, pp. 61-70.

Jobling, C. P. (1991). Building better graphical user interfaces for CACSD - the case for object-oriented programming. *IFAC CAD in Control Syst.*, Swansea, UK, Pergamon, Oxford, UK, pp. 147-151.

Kay, H and B. Kuipers (1993). Numerical behavior envelopes for qualitative models. *Proc. of the 11th Nat. Conf. on Artificial Intelligence (AAAI-93)*, MIT Press, Cambridge, MA, pp. 606-613.

Kuipers, B. (1987). Qualitative simulation as causal explanation. *IEEE Transactions on Systems, Man and Cybernetics*, **17**, 432-444.

Linkens, D. A., et al. (1991). Experiences from a prototype environment for intelligent modelling and simulation. *IFAC CAD in Control Syst.*, Swansea, UK, Pergamon, Oxford, UK, pp. 59-64.

Otter, M. (1992). DSblock: A neutral description of dynamic systems. DLR - German Aerospace Research Establishment, Postfach 1116, 82230 Wessling, Germany

Rahbar, M. T., S. Bennett and D. A. Linkens (1990). Strategies for designing a knowledge based environment for modelling and simulation. *UKSC'90*, UK Simulation Council, Burgess Hill, UK, pp. 146-149.

Takagi, T. and M. Sugeno (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. on Syst., Man, and Cybernetics*, **15**, 116-132.

Tanyi, E. B., D. A. Linkens, S. Bennett (1992). Knowledge acquisition and hierarchical structures for modelling and simulation. In: *Artificial Intelligence, Expert Systems and Symbolic Computing* (E. N. Houstis and J. R. Rice, Eds.), pp. 60-69, Elsevier Science Publishers B V, North-Holland.

SHAPING CACSD FOR PRACTICAL USE IN THE PROCESS INDUSTRY

R. Schumann*, S. Körner*, K. Baker and M. Strickrodt****

* *Fachhochschule Hannover, Fachgebiet Automatisierungstechnik, REPAM, Ricklinger Stadtweg 120, D-30459 Hannover, Germany*
** *University of Glamorgan, Dept. of Mech. and Man. Engineering, Pontypridd, Wales (UK)*

Abstract: During the last decade a variety of academic CACSD tools has been developed which allow the experimental use of computer aided process identification and controller design methods by academic experts. This paper proposes a new approach for the design of an industrial CACSD tool which is tailored to the requirements and abilities of industrial users in the process industry. The approach is based on a standardized CACSD procedure and a process model evolution scheme which simplify the use of CACSD methods under industrial conditions and relieve the industrial nonexpert user from the unnecessary theoretical load of academic CACSD programs.

Key Words: Industrial control; CACSD; identification; controller design

1. INTRODUCTION

Today a large number of CACSD (Computer Aided Control System Design) programs is available on the software market supporting more or less all CACSD phases like process identification and modeling, controller design, system simulation and analysis (Schumann, 1989). However, most programs are of academic origin providing analysis and design methods and tools developed in and for an academic environment. Now, in the process industry controller design tasks have to be solved by the process and control engineer for complex multi-input multi-output (MIMO) processes. Using academic CACSD programs for the solution of these design tasks will lead in general to mathematically complex process models and to the use of powerful theoretical controller design methods which, however, can be understood and handled only by academic control experts - even if the CACSD program is equipped with a sophisticated user guidance system as described in (Meier zu Farwig and Unbehauen, 1991). Moreover, most of the user interfaces of academic CACSD tools were designed to enable extensive tests of various algorithms and methods but do not support efficiently the solution of standard industrial controller design tasks.

This paper presents an industrial CACSD scheme which is tailored to the needs of the control engineer in the process industry. The design is based on numerous discussions with technicians and engineers in the process industry (Bayer, PreussenElektra) and in companies providing process control engineering, equipment and/or systems for this industry (Hartmann&Braun, Siemens). The industrial CACSD scheme is streamlined to support the industrial user on his traditional controller design path. It enables a more efficient and reliable solution for industrial controller design tasks than by purely manual design. The scheme includes:

1. a model evolution scheme for the adaptation of the process model complexity to the practical requirements and
2. the definition of a standardized CACSD procedure.

The paper is organized as follows: In the next section the traditional approach to control system design is outlined as it is still practised in the process industry today. Then the paper focuses on both components of the proposed industrial CACSD scheme, i.e. the model evolution strategy and the standardized CACSD procedure. An extensive design

example illustrating the proposed industrial CACSD scheme will conclude the paper.

2. INDUSTRIAL CONTROL SYSTEM DESIGN

Practical control system design in process industry is mostly based on a rather rough description of the typically MIMO process. An example for this is shown in Fig. 1 where a steam generator is represented by a simple flow chart in which 6 measurement points (process outputs) and 4 manipulation points (process inputs) and 7 PID (Proportional plus Integral plus Differential) control blocks can be detected.

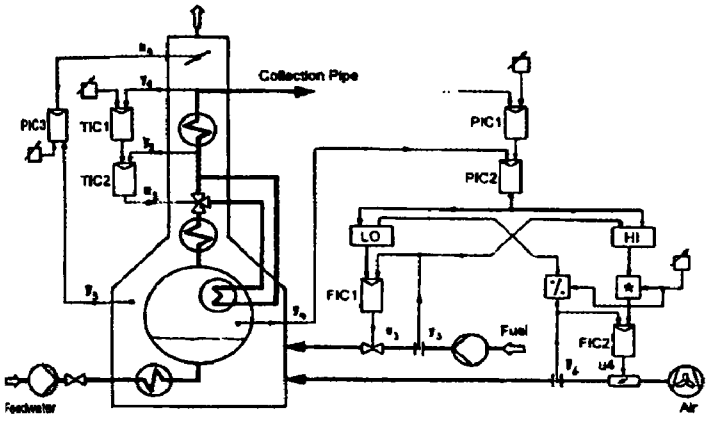


Fig. 1. Steam generator process

2.1. Simple SISO Approach for MIMO Processes

For many technical MIMO processes as for this steam generator standard control schemes (often complicated and nonlinear) are in use which have been developed by generations of process and control engineers in an intuitive way. However, without such a standard control scheme the starting point for practical controller design is in general the simplifying assumption that it is sufficient to split the MIMO process into independent SISO (Single-Input Single-Output) subprocesses by associating each process output to be controlled to the process input with the greatest influence on it. For each of these independent SISO main I/O paths a separate PID controller is implemented on the basis of rather rudimentary process informations like rough estimates for process gain and dominant time constant or possibly (and this is already looked at in industry as advanced time consuming and expensive procedure) based on step response experiments. The separate PID controllers are tuned by human expert knowledge - or better, the experienced industrial engineer just knows how to tune such a process by rules of thumb or by intuitive optimization. The restriction to PID controllers results from the fact that in industry these controllers are still standard. More advanced control algorithms like state controllers, discrete control algorithms or multivariable controllers are usually not available as standard function blocks in industrial controllers - and also not necessary to the understanding of the industrial control engineer because the functionality of these controllers is too complicated and difficult to tune.

2.2. Intuitive Extension of the simple SISO Approach

For 90% of industrial control design tasks the simple SISO approach with separate PID controllers is sufficient. Only if this approach fails due to unacceptable control performance a deeper process analysis is done by intuitive means in the sense that observed changes in process gains and time constants or coupling effects between the SISO subsystems are now taken into account for the controller design in addition. Then the beforehand strictly separated SISO control systems are supplemented with compensating elements to cope for the observed effects. So for the compensation of changing process gains a gain scheduling scheme is often used for the corresponding PID controller and crosscouplings between SISO subsystems are compensated by adding feedforward compensators etc., where all these measures are done more or less in the same intuitive way as for the design of the SISO PID controllers themselves. By time a complicated industrial control scheme may develop as indicated in Fig. 1 for the steam generator which may even contain such nonlinear elements like multiplication and division of signals, min/max-selection etc. and which is difficult to analyze theoretically.

In the next section, an industrial CACSD scheme for industrial controller design is described which is streamlined to the above described approach in the process industry and intended to make it more systematic and transparent.

3. INDUSTRIAL CACSD SCHEME

The proposed industrial CACSD scheme is based on two principles:

1. a model evolution scheme which includes four standard control system structures yielding the simplest solution with acceptable control performance and
2. a standardized CACSD procedure with reduced degrees of freedom with respect to process model and controller structure selection.

3.1. Model Evolution Scheme

The practical design path for industrial control systems as described above is starting with the simplest control system structure, i.e. separated SISO subprocesses controlled by individual PID controllers and extended to more complicated control schemes for compensation of process nonlinearities or coupling effects only in case the simple solution does not work sufficiently. The proposed industrial CACSD scheme follows this principle by defining a

model evolution scheme comprising four different control system structures as shown in Fig. 2 and described in the following.

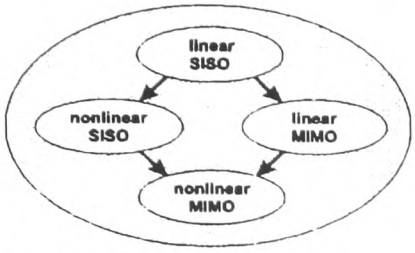


Fig. 2. Model evolution scheme

Linear SISO model. The first attempt for the controller design is based on the Linear SISO model which assumes that it is sufficient to represent the process by a model with separate linear SISO submodel blocks as shown in Fig. 3.

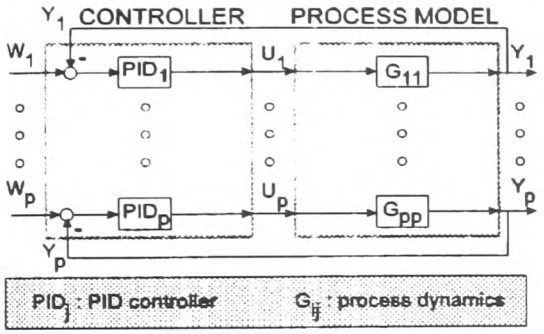


Fig. 3. Linear SISO process model and controller

The selection of the SISO main I/O paths is done by associating every process output to be controlled to the process input with the greatest influence on it. For each SISO submodel an independent linear PID controller is designed, see Fig. 3. Only in case that no acceptable control behaviour can be achieved using the standardized CACSD procedure as described in the next section the system structure is extended depending upon the observed unacceptable effects to one of the following alternatives.

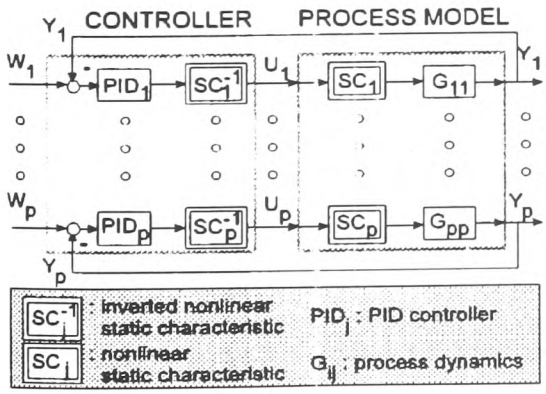


Fig. 4. Nonlinear SISO process model and controller

Nonlinear SISO model. In case that control problems are detected to be related with varying process gains, the Linear SISO model should be augmented to the Nonlinear SISO model. This model is combined from separate SISO submodels for the main I/O paths in form of simple Hammerstein (alternatively also Wiener) models each with a linear dynamic and a nonlinear static part, see Fig. 4. For each nonlinear SISO submodel a complementary nonlinear controller is designed with a nonlinear static block for compensating the submodel's nonlinearity and a linear PID controller tuned for the submodel's linear part, see Fig. 4.

Linear MIMO model. In case that primarily coupling effects deteriorate the control performance, the Linear SISO model should be extended to the Linear MIMO model reflecting also the crosscoupling effects between process inputs and outputs by additional linear coupling blocks, see Fig. 5.

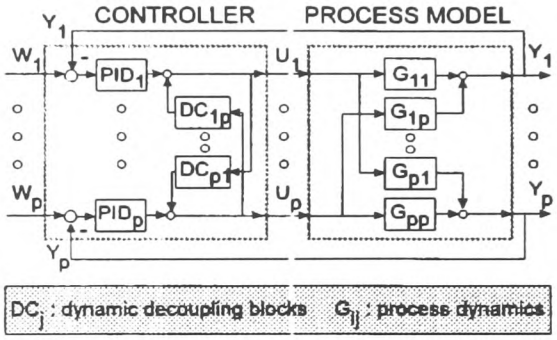


Fig. 5. Linear MIMO process model and controller

The corresponding standard controller structure contains singlevariable PID controllers for the main I/O paths as in the Linear SISO case, which are complemented with feedforward controllers designed to compensate for the effects of the linear dynamic coupling blocks of the process model.

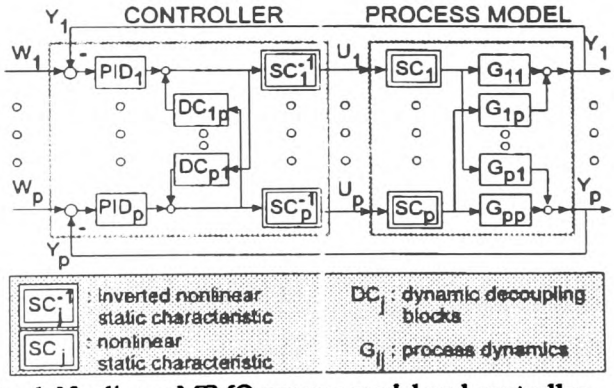


Fig. 6. Nonlinear MIMO process model and controller

Nonlinear MIMO model. Only in case that none of these alternative process model structures yields sufficient control behaviour the Nonlinear MIMO model may be tried as the most complicated model structure in the proposed

model evolution scheme. This model structure can be gained either by supplementing the Nonlinear SISO model with nonlinear coupling I/O paths (each of which containing a linear dynamic and a nonlinear static block) or by extending the Linear MIMO block by nonlinear static blocks at each input (or output) of the model as shown in Fig. 6. In this case the corresponding controller structure is defined by extending the controller structure of the Linear MIMO model by nonlinear static compensation blocks at the process inputs, see Fig. 6.

3.2. Standardized CACSD Procedure

The standardized CACSD procedure is applicable to each of the above described models and described here for the case that the process model is generated by process identification from experimental data. The CACSD procedure can be split in three main CACSD phases, see Fig. 7.

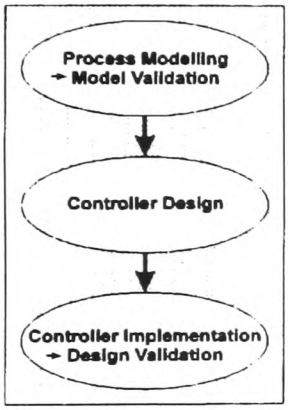


Fig. 7: Standardized CACSD Procedure

PHASE I: Process Modelling. As first step in the standardized CACSD procedure the process model is to be generated in the computer from experimental data. For the above specified 4 process model structures only two different CACSD tools are required:

1. an identification tool for generation of a linear dynamic SISO or MISO model from experimental data and
2. an identification tool for the determination of static characteristics in the SISO and MIMO case.

In case of the linear models the application of the first CACSD tool will lead directly to the required model. In case of the nonlinear models the static characteristics have to be determined first making use of the second CACSD tool; then by precompensating the model nonlinearities by their respective inverse the linear dynamic blocks are identifiable using the first CACSD tool. For validation of the process models graphical inspection is proposed allowing also an inexperienced user the detection of bad models by comparison of experimental and simulated data (in the future also other 'quality' measures will be used). In case that no good correspondence between experimental and simulated data can be achieved with the used process model structure Phase I of the CACSD procedure has to be repeated with the next more complex process model structure.

PHASE II: Controller Design: As shown in Fig. 3 to Fig. 6 the process model structure found in Phase I is directly reflected in the associated control system combined from:

1. linear single variable PID controller blocks tuned for the linear dynamic part of the associated main I/O path
2. nonlinear static blocks defined as inverse blocks of the corresponding process model nonlinearities and/or
3. linear feedforward compensating blocks tuned to reduce effectively the crosscoupling effects.

The tuning of the linear PID controllers and the feedforward compensating blocks can be done easily by numerical optimization in appropriately separated control subsystem. The predicted control system performance is checked by simulation of the complete control system with process model and controller, however, as the complete control system has been designed to cope only for the modeled effects only direct design errors can be detected which may accordingly be corrected by just repeating Phase II.

PHASE III: Controller Implementation: The implementation of the designed control system is a nontrivial task not only due to possibly unmodelled process model parts but also due to potential differences between the controller elements used in the simulation in Phase II and the ones really applied to the process with industrial control systems (modified PID algorithms, limiters, anti windup schemes etc.) which have to be taken into account. The crucial validation of the complete controller design is thus based on the comparison of the real control performance with the simulated one reached in Phase II. In case that the control performance at the real process is not sufficient and differs obviously from the simulated one the standardized CACSD procedure has to be repeated from Phase I with the next more complex process model structure.

4. PROTOTYPE REALIZATION OF THE INDUSTRIAL CACSD SCHEME

4.1. Nonlinear MIMO Pilot Plant

The scheme of the nonlinear two input two output pilot plant is shown in Fig. 8. Its main component is a semi-closed water tank filled with water by the waterpump (No.1 of Fig. 8) and with air by the airpump (No.3 of Fig. 8). Two valves, one for water level and one for air pressure (No.2 and 4) allow to adjust the operating point of this plant. Valves 5 and 6 provide the means to generate de-

fined disturbances for the controlled variables water level Y_w and air pressure Y_a, which are manipulated by the control voltages U_w for the waterpump and U_a for the airpump. The plant is clearly crosscoupled in the sense that the waterpump does not only influence the water level, but also the air pressure and vice versa the air pump does also influence the water level.

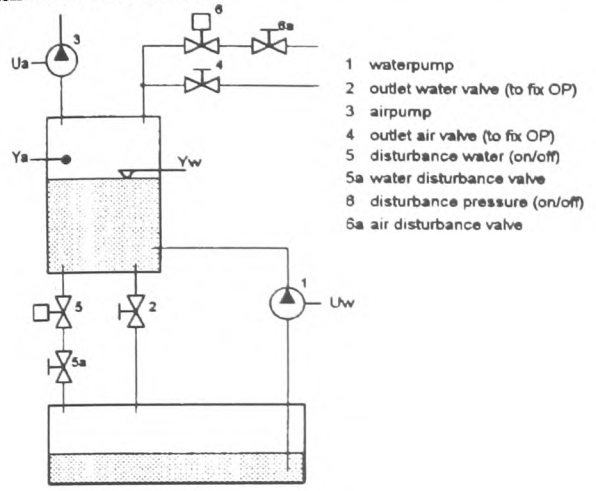


Fig. 8. Nonlinear MIMO pilot plant

4.2. Utilized Tools and Methods

As already pointed out numerous academic CACSD tools are available on the market (Schumann, 1989; Schmid, 1993; Frederick et al., 1992) but unfortunately there was no single tool available at the time of the prototype realization which was suited to support all tasks of the standardized CACSD procedure appropriately for an industrial environment. Thus a patchwork of tools had to be selected for the prototype realization, see Table 1.

Table 1 Utilized CACSD tools

Task within Standardized CACSD Procedure	CACSD tool
identification of linear dynamics	CADACS by University of Bochum
identification of static characteristics	*EASYSTAT* by FH Hannover
simulation and controller design	DORA 5.1 / DORA-Fuzzy by University of Dortmund
controller implementation validation	TCS with Loopdraw by EUROTHERM
simulation and presentation	SIMULINK / MATLAB by Mathworks

Identification of linear dynamics. CADACS with its real-time module for identification experiments provides various process identification methods. For the prototype realization the simple and reliable Moncman method was chosen which determines an n'th-order-lag-approximation model from a measured step response.

Identification of static characteristics. No commercial CACSD tool was found which supports this work effectively. So a proprietary CACSD tool, EASYSTAT, was created to deal with the time-consuming job of investigating steady state characteristics of SISO and MIMO processes. EASYSTAT allows automatic determination of static single and multidimensional characteristics in open and closed loop.

Simulation and controller optimisation. For this part DORA was chosen because especially the simulation part DORA-Fuzzy offers integrated, simple and efficient optimization facilities which allow PID parameter tuning in a block oriented environment. The PID controllers were optimized for setpoint changes using a quadratic controller design criterion, which balances control performance and actuator effort. For the decoupling feedforward controllers, standard lead/lag blocks were numerically optimized to reduce the coupling effects between the main I/O paths. The nonlinear characteristics of the control system were realized as inverse of the identified process nonlinearities using look-up tables.

Controller implementation and on-line validation. As typical industrial controller device the TCS (Turnbull Control Systems) 6370 controller was chosen as target system for the designed controller structure. The control system was implemented using the graphical blockoriented configuration software LOOPDRAW providing the means to realize nonlinear multivariable controller structures using linear dynamic blocks and static nonlinear blocks (as look-up tables).

4.3. Experimental Results

Now the complete industrial CACSD scheme will be illustrated by experimental results of the prototype implementation. Among the many results a ramped set point change (as normally applied in process industry) on the water level was chosen to demonstrate the overall performance of the designed control systems. For this purpose the process input variables water and air pump voltage U_w and U_a, as well as the process output variables water level Y_w and air pressure Y_a were recorded for the different cases

Linear SISO model. The application of the described standardized CACSD procedure produced the simulation and real time control results shown in Fig. 9. The control behaviour at the real plant shows oscillations and crosscouplings (which were expected) which the simulation does not show at all. The significant difference is obviously due to the fact that the simulated model does not cover any nonlinear or coupling effect. The observed differences and the poor control behaviour indicate that the model complexity is not sufficient.

Nonlinear SISO model. This was the first try to reduce the observed oscillation effects in the Linear SISO case. The simulated control system behaviour is identical to the linear SISO case (besides a change in the control signal scales due to the transfer of the process gains from the linear dynamic to the nonlinear blocks). However the real control performance becomes much better with respect to the oscillation effects observed in the first try, see Fig. 9. This is obviously due to the modeling and compensation of the process nonlinearities. Nevertheless the differences between simulation and real time results indicate still the existence of unmodelled parts in the process model and the observed real time control behaviour was not accepted.

Nonlinear MIMO model. To improve the results of the nonlinear SISO case the nonlinear MIMO model was tried (The linear MIMO approach was omitted due to the obvious existence of model nonlinearities). The comparison of the simulated control system with the realtime control experiments showed a much better coherence than in the first two cases. Furthermore, the crosscoupling effects are clearly reduced compared to the simpler process models, see Fig. 9. So the overall performance was accepted and the industrial CACSD scheme came to a successful end.

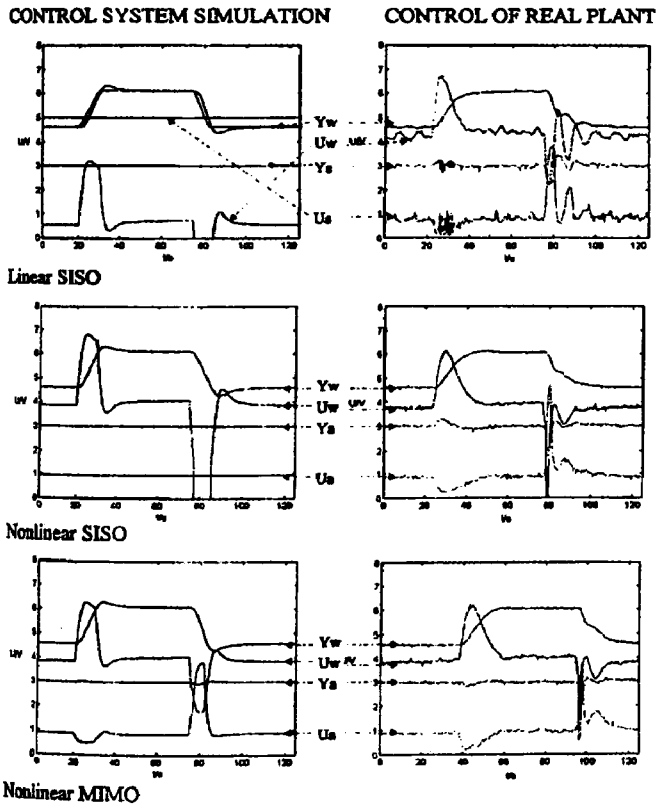


Fig. 9. Comparision of simulated and real process behaviour

5. SUMMARY (CACSD ASPECTS)

The proposed industrial CACSD scheme was designed for the solution of practical controller design tasks in the process industry. The combination of a standard model evolution scheme (from linear SISO to nonlinear MIMO) with a standardized CACSD scheme (including process identification, controller design and implementation) simplifies the CACSD procedure for the industrial user and allows a simple adaptation of the control system complexity to the practical requirements. The prototype application to a laboratory plant demonstrates the principal feasibility of this approach. Other applications, e.g. to climate plants, have shown similar results. However, it is clear that the proposed industrial CACSD scheme in its basic version has its limits with respect to the used model structures which have been selected to support rather the practized industrial design process than to fulfil theoretical conditions. So, future work will concentrate on the refinement of the scheme with respect to the use of alternative process model structures and an early detection of undermodeling. Also the use of alternative control structures oriented at more refined process model structures will possibly require other tuning procedures. Moreover, the prototype realization using a variety of different CACSD tools has to be replaced in the future by a new industrial CACSD tool realizing efficiently the outlined industrial CACSD scheme with a user interface designed for the industrial user.

6. ACKNOWLEDGEMENT

The research was carried out with support of the DAAD through the Anglo-German Academic Research Collaboration Programme ARC, Project 313-ARC-VII-93/86/scu, for the collaboration with the University of Glamorgan, Project 520. Further this research was partly supported by the Volkswagen foundation, project 210-70631/9-38-1/92.

7. REFERENCES.

Frederick, D K;Herget, C J; Kool, R; Rimvall, M, (1992), ELCS - The extended list of control software, *Dept. of Automatic Control*, Swiss Federal Institute of Technology (ETH), Zürich.

Meier zu Farwig, H and Unbehauen, H (1991). Knowledge-based system identification. *IFAC Symposium on Identification and System Parameter Estimation*, Budapest, Hungary.

Schmid, C (1993), Verfügbare Softwaretools Seminar "Rechnergestützte Analyse und Entwurf von Regelungssystemen", *VDI Seminarhandbuch, VDI-Bildungswerk, Düsseldorf.*

Schumann, R (1989). CAE von Regelsystemen mit IBM-kompatiblen Personal Computern. Automatisierungstechnische Praxis (atp), Vol. 31, No. 8, 349 -359.

KNOWLEDGE ACQUISITION AS PART OF A PRACTICAL CACSD APPROACH

M. Strickrodt*, R. Schumann** and K. J. Baker*

Introduction

This paper describes an approach to process modelling that aims at making use of the experience with respect to process behaviour that engineers in industry accumulate during their everyday work. Making extensive use of graphical interface facilities, the approach is implemented into an interactive computer program which is designed in such a way that it can be handled by engineers without experience in process modelling. In the overall context of Computer Aided Control System Design (CACSD), the suggested approach, called *MODEL*^{ing}, will serve as a preprocessor to the modules 'Experimental Process Identification', 'Controller Design' and 'Process Simulation', which are being developed within the collaborative research project between the University of Glamorgan and the Fachhochschule Hannover. These modules will likewise be geared at industrial users with only basic control engineering knowledge.

The need for such simplified approaches results from the fact that control engineering programs (cf. review by Schumann[1]) have still not found a widespread application in manufacturing and process industry outside the control engineering departments which normally only exist in bigger companies. However, even in such companies, which do have specialists departments, the valuable practical experience of the area engineers is not usually exploited in a systematic way.

The overall structure of the suggested approach, which is still being developed, is broadly outlined in this contribution while one of its features, the aggregation of a list of descriptive attributes, will be considered in some more detail.

Unlike most of the previous work on intelligent modelling, which focused on the theoretical component analysis of processes, combined with an implementation of the 'deep' knowledge (e.g. physical laws) that underlies each of these components, the approach introduced in this paper focuses on the experience of practitioners, which is also known as 'shallow' knowledge. The earlier work, most notably the extensive research carried out in the Department of Automatic Control and Systems Engineering at the University of Sheffield, which has mainly been published between 1987 and 1993 (Linkens, et al.[2]; Xia, et al.[3]; Tanyi, et al.[4], to name a few), is therefore complemented by this approach.

The Overall Structure Of The Approach

The particular task of *MODEL*^{ing} as a preprocessor is the collection and provision of 'a priori' knowledge in control engineering terms - in particular for the 'Experimental Process Identification' module - which can have different formats, depending on the kind and detail level of information supplied by the area engineer: Ideally, the largely unstructured knowledge of the practitioner is translated through the *MODEL*^{ing} program into a coarsely parameterised transfer function or a nonlinear fuzzy hybrid model, which has been proposed by the authors. Apart from the possibility of other, 'intermediate' models, the preprocessor should at least provide a list of descriptive attributes related to the process considered (Figure 1).

As the potentially useful information can be quite diverse, an optimised structural layout of the knowledge acquisition procedure is of major importance for keeping the required effort to a minimum. The Object

* University of Glamorgan, Department of Mech. & Man. Engineering, Pontypridd, Mid Glamorgan, UK
** Fachhochschule Hannover, Fachgebiet Automatisierungstechnik, REPAM, Hannover, Germany

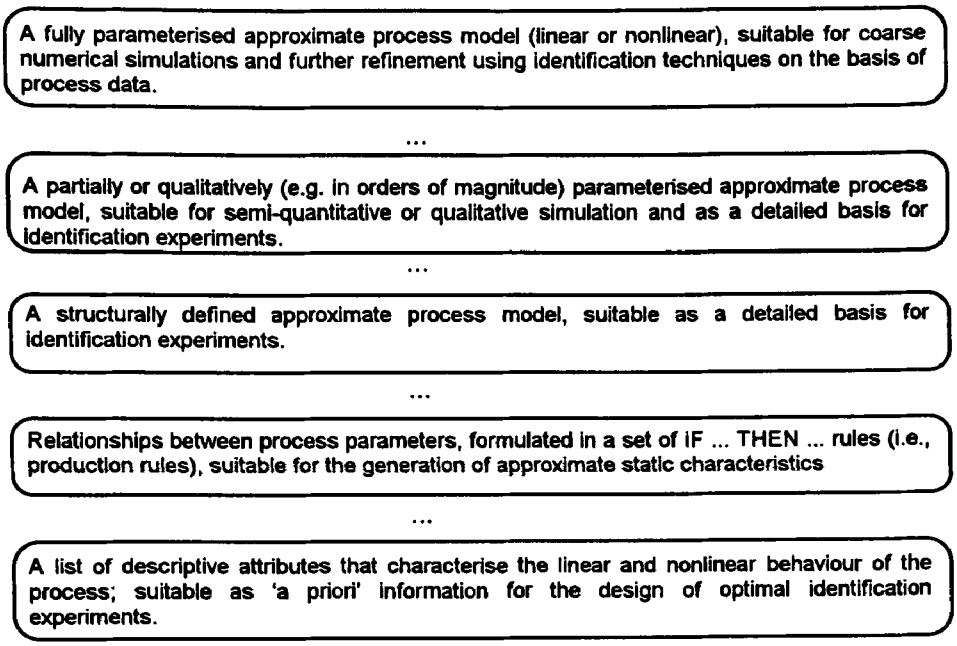


Figure 1: The Range Of Possible Results From Applying The *MODEL*^{ing} Approach
(The dots between the levels of abstraction indicate that intermediate model definitions are possible)

Modelling Technique (OMT) by Rumbaugh et al.[5] was chosen as a structured approach to analysis and design because of its flexibility combined with notational simplicity. Among the three types of models within OMT, the dynamic model, which represents the behavioural and control aspects of a system, is of primary interest for the given problem.

Figure 2 shows the top-level dynamic model of the *MODEL*^{ing} approach which represents the global view at the sequential arrangement of the main modules within the approach. The heuristic modelling step, which is most important for the envisaged application of the approach, is shown with its main sub-sequences: The nonlinear Multiple Input / Multiple Output (MIMO) modelling approach is split into repetitive sequences of Multiple Input / Single Output (MISO) modelling, with the number of repetitions according to the number of outputs. The MISO modelling in turn is substructured - mainly into repetitive SISO modelling of linear dynamics (cf. Strickrodt et al.[6]) and the modelling of nonlinear static relationships. The latter sub-state within the heuristic modelling approach - the modelling of nonlinear static relationships - is further sub-split, mainly into rule-based modelling and an approach to acquire a list of attributes that refer to the static characteristic of the modelled component.

An Aspect Of The Proposed Approach: The Attributes List

Rather than trying to determine standard types of static characteristics (like backlash, friction, or hysteresis), the basic idea of the attributes list aims at checking for attributes that would appropriately be associated with the shape of the static characteristic. The main attributes for static characteristics are:

- single / multiple valued, multiple variable
- time invariant / time variant
- continuous / discontinuous
- boundless / saturating; upper and/or lower bound
- positive and/or negative slope of static characteristic
- static characteristic with absolute or relative maxima and/or minima
- intermediate 'plateau' of constant output level over input range (intermediate unsensitivity)

Beyond the Boolean yes/no-decision as to whether an attribute would be appropriate for the description of the considered static characteristic, the *MODEL*^{ing} approach tries to elicit further, in particular numerical, information from the user. The advantage of this approach is its modularity and the flexibility with respect to addressing combinations of different nonlinear effects.

Such an attributes list with information on nonlinearities greatly simplifies the design of identification experiments in that it helps either to avoid critical areas of the operation range, or to run specific tests to gain more information about the nonlinearities. For the acquisition of attributes and numerical information, a combined query and form-filling approach is applied, in which questions about typical behaviour characteristics (i.e., *effects*) are answered and the related process attributes are passed to the appropriate slot within the frame-based knowledge representation.

From an object-oriented point of view, the nonlinear model on the basis of an attributes list is treated as a separate object that is associated with other parts (i.e., objects) of the model, like linear dynamic SISO components. The advantages of object-orientation in CACSD are manifold (cf. Jobling, et al.[7]). The decision to store the information on the model in several associated objects results here in a modular and flexible, frame-based knowledge representation.

While the list of attributes can be printed or directly passed as part of the 'a priori' knowledge to the process identification module, the numerical information is additionally translated to a fuzzy model to allow for the plot of a first, coarse static characteristic. Fuzzy interpolation, using defaults like triangular input membership functions with $\sum\mu_i = 1$, singletons as output membership functions and multiplication as implication method has been found most versatile, in particular for multiple valued static characteristics.

Conclusions

The analysis of CACSD programs and their application in industry indicated the need for new approaches that emphasise practical support. The approach that has broadly been outlined is a step in this direction: it enables practising process engineers to apply their experience to process modelling for control engineering purposes. The results of applying this knowledge acquisition approach are passed in the format of partially defined process models and supplementary information to the other program modules that will form part of the envisaged new overall approach. All modules will be provided with interfaces to some commercially available CACSD programs so that analyses, simulations and optimisations beyond the scope of the new practical approaches can be carried out by experienced control engineers.

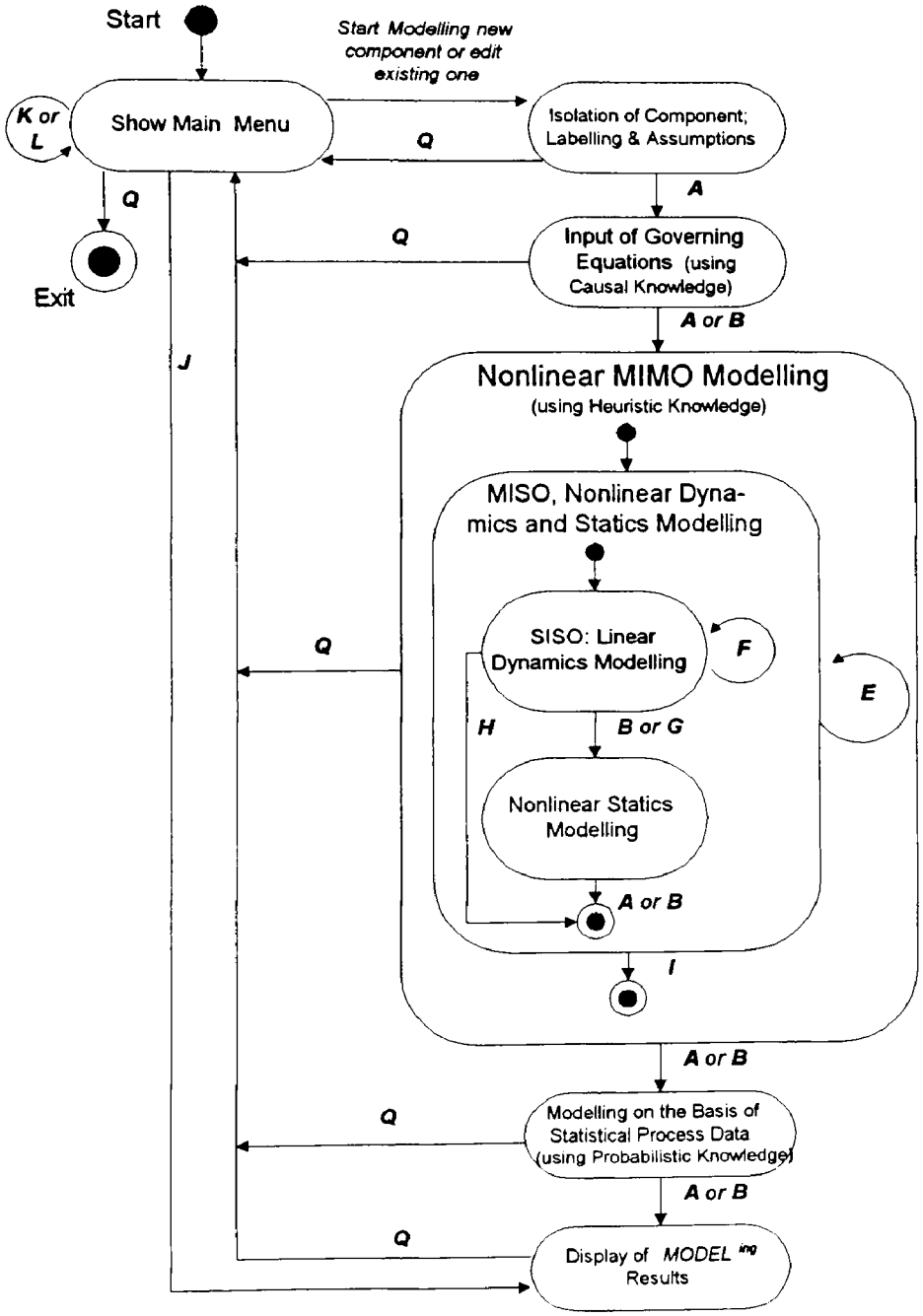
Acknowledgements

The authors would like to thank Mr S. Körner for the discussions. The research was carried out with the support of both the British Council and DAAD (Deutscher Akademischer Austauschdienst) through the Anglo-German Academic Research Collaboration Programme ARC, Project No. 520 and 313-ARC-VII-93/86/scu, respectively, for the collaboration between the University of Glamorgan and the Fachhochschule Hannover.

References

1. Schumann, R: "CAE von Regelungssystemen" (in german), atp - Automatisierungstechnische Praxis, R. Oldenbourg Verlag, vol. 36, no. 3, pp. 51-59, 1994
2. Linkens, D A; Xia, S; Bennett, S: "A computer-aided qualitative modelling and analysis environment using unified principles (QREMS)", Int. Conf. on Bond Graph modell. ICBGM'93, SCS, CA, USA, pp. 53-58, 1993
3. Xia, S; Linkens, D A; Bennett, S: "Integration of qualitative reasoning and Bond graphs: an engineering approach", IMACS Trans. Special Vol. on Bond Graphs for Engineers (eds.: P C Breedveld and G. Dauphin-Tanguy), Elsevier Science Publishers B.V., North-Holland, pp. 323-332, 1992

4. Tanyi, E B; Linkens, D A; Bennett, S: "Knowledge acquisition and hierarchical structures for modelling and simulation", Artificial Intelligence, Expert Systems and Symbolic Computing (E. N. Houstis and J. R. Rice, Eds.), Elsevier Science Publishers B.V., North-Holland, pp. 60-69, 1992
5. Rumbaugh, J, et al.: "Object-oriented modeling and design", Prentice-Hall, Englewood Cliffs, NJ, 1991
6. Strickrodt, M; Schumann, R; Baker, K: "An integrated knowledge engineering approach to process modelling for CACSD", to appear in Proc. of IFAC'96 World Congress, 1996
7. Jobling, C P; Grant, P W; Barker, H A; Townsend, P: "Object-oriented programming in control system design: a survey, Automatica, Pergamon Press, vol. 30, no. 8, pp. 1221-1261, 1994



EVENTS:

A: finished modelling step: (partial) completion of requested information

B: user skipped modelling step (interpretation: "don't know")

NOTE : the following events *C* and *D* are implicitly attached to every modelling step (and sub-step)

C: user request to edit last modelling step / re-enter the preceding state

D: given erroneous input / post error message and re-enter the preceding state

recurrent:
E: finished modelling step [n<number of outputs] / n=n+1

recurrent:
F: finished modelling step for the main input-output relation [i<number of combinations of distinguished settings of the other inputs, i.e., influences] / i=i+1, next influence vector

G: finished modelling step for the main input-output relation [i=number of combinations of distinguished settings of influences and the modelled process has a steady state]

H: finished modelling step for the main input-output relation [i=number of combinations of distinguished settings of influences and the modelled process has no steady state]

I: finished modelling step [n=number of outputs]

J: request to display results

K: selection of menu item 'delete' / delete a component

L: selection of menu item 'code generation' [sufficiently defined model] / generate simulation code

Q: selection of menu item 'Quit'

Figure 2: The Top-Level Dynamic Model Of The *MODEL*^{ing} Approach
(in simplified OMT-notation: system states in rounded boxes; events trigger the transition between states; the arrows with *event*-annotations mark transitions [conditions of transition] / associated actions)

A Fuzzy Hybrid Model For The Simulation Of Nonlinear Dynamic Processes

for publication in the IEEE Transactions on Systems, Man, and Cybernetics

Matthias Strickrodt and Keith J. Baker[i]

Abstract -- This paper discusses a new, simplified approach to the modelling of multivariable nonlinear dynamic processes. After pointing out the need for such modelling approaches and indicating the merit of applying a fuzzy hybrid concept for this purpose, an overview of the previous work is given. In the stepwise modelling sequence, which is illustrated with an example, some default settings are introduced. These defaults play an important role in the simplification and standardisation of the suggested modelling approach. After the discussion of some aspects and particular advantages of the approach, its applicability to a real process is demonstrated.

While the highly successful approach of Takagi and Sugeno is advantageous for the combination of static system equations, it is argued that the presented approach which is particularly aimed at combining locally valid dynamic system equations to a nonlinear, global process model is more suitable for dealing with dynamic processes.

1. Introduction

Whilst, in recent years, there have been considerable developments in modelling and simulation for control engineering purposes, these have not been widely applied in process and manufacturing industry. Future research will need to focus on industrial applicability if the benefits are to be realised in industry. For the development of practically applicable yet progressive approaches, it is especially important to understand the particular constraints of modelling and simulation in industry: Potential users of modelling approaches in most parts of manufacturing industry are relatively inexperienced in modelling and simulation, and in any case they are timewise extremely constrained.

Manuscript received . . .
[i] The authors are with the Department of Mechanical and Manufacturing Engineering, University of Glamorgan, Pontypridd, Mid Glamorgan, CF37 1DL, U.K.
The research was carried out with the support of the British Council through the Anglo-German Academic Research Collaboration Programme ARC, Project 520, for the collaboration with the Fachhochschule Hannover.

Theoretical process modelling is normally not possible in an industrial environment due to the extensive time consumption, the required expertise and the high complexity of production processes. The use of process models is therefore normally restricted to very specific conditions and operating points: basic process identification approaches that are applied in industrial environments yield simplified, linear single input / single output (SISO) transfer functions that are only locally valid. Although approaches for the identification of nonlinear conventional [1] as well as, for example, fuzzy logic [2] or neural networks [3] based process models exist, they are quite complex and require a good understanding in these fields, which is not yet widespread in most parts of industry.

Built specifically on the presupposition of 'patchy' process knowledge in the form of the above mentioned locally valid, piecewise linear SISO models, the formalism presented in this paper addresses the industrial need and enables a simplified approach to the modelling of highly nonlinear, multivariable global process models. The local models are either obtained through simple local process identification experiments or derived from the practitioner's experience, using an 'intelligent' interface [4, 5]. This particular kind of basic information that is required for the approach is a unique feature of the proposal as is its simplicity. Furthermore, the proposed approach is particularly aimed at continuous processes whose dynamic characteristics vary, depending on one or more parameters, which is frequently the case in process industry. This type of truly 'nonlinear dynamic modelling' addresses therefore significantly more complex situations than the commonplace understanding of 'nonlinear (and) dynamic modelling', where the dynamic behaviour as such is linear and only preceded (or succeeded) by a nonlinear static characteristic (e.g., the simplified Hammerstein and Wiener-models, respectively [6, 7]).

The approach is based on fuzzy logic, which is used as an efficient means to integrate the local equations and to adapt the dynamic process behaviour continuously according to varying values of the influencing parameters. The fuzzy notion was chosen especially because of its capabilities to model very nonlinear relationships. It is also advantageous compared with purely data-driven approaches like neural networks in domains where either data is not always available, existing process knowledge should be used, or the modelling result must be interpretable. All these constraints apply to the situation targeted with the proposed approach.

The association between input and output variables in terms of plain fuzzy rules of the kind "IF *A* is *small* AND *B* is *big* THEN *C* is *medium*" is a purely static relationship; a fuzzy controller (or model) can therefore be fully represented by a static characteristic in form of one or more surface plots that relate the output directly to the inputs. Such surface plots are a standard viewing option in virtually all fuzzy modelling tools. Nevertheless, it is possible to model continuous dynamic processes using the basic fuzzy modelling approach with higher order derivatives (discrete: z^{-n}, $n \geq 1$) as additional inputs. This kind of dynamic modelling has, however, its drawbacks:

- The required rules become quite complex and cannot be formulated on the basis of experience anymore - only the identification of the fuzzy model using measured data is feasible.
- Analyses (e.g. stability) are hardly possible.
- Very small discrete simulation increments are necessary in order to achieve satisfactory continuous effects.
- The simulation step size influences the response of the model.
- In industry, the acceptance of purely fuzzy-based systems is low.

Another limitation of the basic fuzzy modelling approach is the saturation effect of the output values at the borders of the specified operating range: the output *C*, for example, is not extrapolated beyond the numerical values associated with the fuzzy sets "MAX." or "MIN."

However, the above limitations and drawbacks which are of importance for the envisaged application can be overcome by combining the fuzzy notion with linear system equations to a fuzzy hybrid system. Another important advantage of such combined systems is the possibility of making use of the various traditional system analysis techniques.

The use of fuzzy-based process modelling is still very uncommon in control engineering compared with its direct application to linguistically defined, model-free control systems, although modelling was already part of Zadeh's early ideas published on fuzzy theory, as Sugeno and Yasukawa [8] point out. While the direct implementation of fuzzy controllers has proved to be successful, the model-based design of control systems becomes necessary in the case of very nonlinear, complex processes [9]. Also, fuzzy process models could be very useful for model-based feed forward control or general system analysis and simulation.

2. Modelling On The Basis Of Partial Information - An Overview Of The Previous Work

A variety of modelling and simulation approaches exist which make use of partial or incomplete process information. Apart from different aspects of their performance, these approaches largely differ in terms of the kind and amount of information to be provided in the modelling process[ii]:

For modelling on the basis of Automata or Petri-Nets (Lunze [11, 12, 13]), for example, the sets of possible system states and input settings, an initial condition and an input sequence must be specified along with the transition function (for Automata) or net topology (for Petri-nets). Both approaches focus on the availability of only coarse (qualitative) measurements of

[ii] for a more comprehensive review please refer to Strickrodt and Baker [10]

the system states as the main source of 'incompleteness' or uncertainty and are applied to the discrete simulation of both continuous and discrete processes.

Qualitative reasoning-based approaches, epitomised by QSIM (Kuipers [14, 15]), are aimed at a very different modelling situation: they require information about the physical equations that are related to the process to be modelled and accommodate coarse, qualitative knowledge with respect to the involved parameters. These equations are linear or nonlinear and generally applicable ('globally valid'), independently of operating conditions or even the particular modelling situation (i.e., based on fundamental causal relationships, also called 'deep knowledge').

The fuzzy hybrid approach that is proposed in this paper, however, aims at modelling situations where only simple dynamic 'black-box' models are available or can be derived. These 'black-box' models are normally only 'locally' valid and linearised: they describe the behaviour of the considered process under very specific conditions with respect to input and influence parameters without any reference to the underlying causal relationships. 'Black-box' models result often from identification experiments and are commonplace in practical (especially industrial) situations where a theoretical analysis of processes in terms of their underlying causal relationships is not possible. An approach to derive locally valid 'black-box' models also from practical experience is suggested in [4]. Using the proposed fuzzy hybrid modelling approach, the simple, locally valid single input / single output 'black-box' models can be combined to a globally applicable, statically and dynamically nonlinear, multivariable process model.

The idea, to make best use of both fuzzy and conventional mathematical approaches, is not new. In the remainder of this section, existing fuzzy hybrid approaches are briefly summarised:

In order to analyse the effects of uncertainty with respect to the parameters of ordinary linear mathematical transfer functions or state space models, the fuzzy notion is applied for example by Jain [16], Grobbelaar [17] and Kandel [18], Rouhani/Tse [19], respectively. The application of fuzzy sets as an uncertainty measure necessitates the definition of fuzzy operators analogous to arithmetic operators on rational numbers because the uncertainties have to be carried through the numeric simulation and the effects of several uncertain transfer function parameters must be combined, increasing the overall uncertainty in the simulation result. This approach is mainly based on replacing the ordinary, crisp parameters by fuzzy numbers according to the fuzzy extension principle [20], and therefore it does not require a rule-base. The uncertainty in the simulation result is represented by a set of system trajectories with different probability annotations. The response to a crisp input is therefore a fuzzy signal (Fig. 1). Although this is an interesting and important application of fuzzy-hybrid systems, it does not take advantage of the nonlinear modelling capabilities of the fuzzy approach.

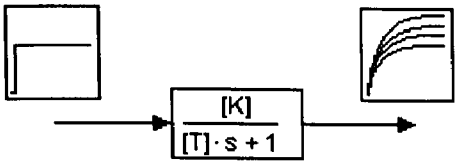


Fig. 1: Fuzzy Hybrid Model to Express Uncertainty in the Parameters - the Square Brackets indicate here Fuzzy Parameters.

The fuzzy hybrid approach suggested by Takagi and Sugeno [2] has a different motivation, which is very similar to ours: it aims at modelling highly nonlinear processes in a simplified manner using piecewise linear mathematical equations which are combined via the fuzzy notion (Fig. 2).

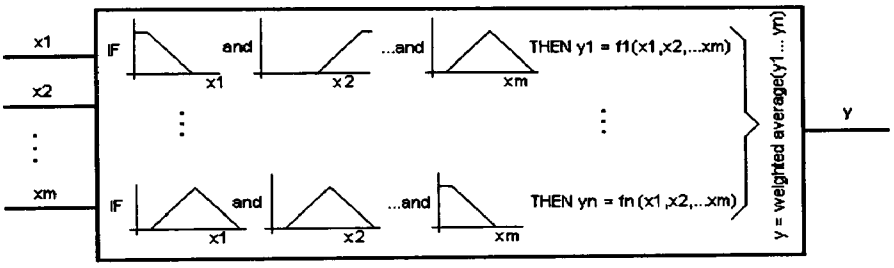


Fig. 2: The Takagi-Sugeno Fuzzy Hybrid Model

Both input and output of these process models, which are aimed at the model-based design of multivariable fuzzy control systems, are non-fuzzy - or 'crisp'. The particular characteristic of this approach is the format of the fuzzy implication where the "THEN"- part of the rules does not assign a specific fuzzy set from the output space to the output variable as is normally the case but defines the output variable in terms of a function of the input variables. The general format of a fuzzy multiple input - single output process law (analogous to a fuzzy control rule but **applied to modelling**) is defined as follows:

$$L^i: \quad \text{IF } x_1 \text{ is } A^i_1,\ x_2 \text{ is } A^i_2,\ \ldots\ x_m \text{ is } A^i_m,\ \text{THEN } y^i = c^i_0 + c^i_1 x_1 + c^i_2 x_2 + \ldots + c^i_m x_m$$

with L^i denoting the i-th process law, c^i_k coefficients, A^i_k fuzzy sets, x_k input variables and y^i the output from the i-th process law. Using the truth value w^i of the premise of the i-th process law, which is calculated as

$$w^i = \prod_{k=1}^{m} A^i_k(x0_k) \quad ,$$

a given input $x0 = (x0_1, x0_2, \ldots, x0_m)$ yields the overall output

$$y0 = \frac{\sum_i w^i y^i}{\sum_i w^i} \quad .$$

The overall output of the fuzzy model is therefore the weighted average of the y^i's.

This approach, which allows for highly nonlinear modelling despite the small number of rules needed, is widely applied and acknowledged, with the MATLAB™ fuzzy toolbox probably being one of its latest implementations. The modelling approach was repeatedly shown to be advantageous in conjunction with fuzzy identification [2, 9] and applied, for example, to helicopter flight control [21].

Compared with other rule-based approaches, a particular strength of the Takagi-Sugeno approach is that normally only very few antecedents among all possible input-combinations are required, which reduces the size of the rule base significantly.

The Takagi-Sugeno model has, however, an important limitation: combining piecewise dynamic system equations, for example transfer functions, differential or difference equations in a similar fashion as the linear static equations above is not generally possible. Although Sugeno and Kang [9] have actually applied this modelling approach for the combination of first and second order difference equations, which are valid under certain process conditions, this is not **generally** permissible. Using dynamic system equations in the consequent - or 'THEN' - part of the Takagi-Sugeno model can lead to erroneous results whenever oscillating signals occur - either through system equations with complex poles or due to a frequency input signal. Since the overall output of the system is determined by averaging trajectories, phase differences lead, for example, to cancellation effects and therefore spurious predictions (cf. section 3.3).

Kuipers and Aström [22] describe a heterogeneous control system that switches between different local control laws[iii] using the fuzzy notion to achieve smooth transitions between adjacent regions. The global heterogeneous control law in this approach is defined as the weighted average of the local control laws, where the weights are returned by the fuzzy membership functions. This concept is therefore very similar to the Takagi-Sugeno model, the main difference being the separation between the fuzzy part, which is responsible for the soft transitions by determining weighting factors, and the actual control laws as well as the aggregation of the separate local outputs (Fig. 3). Due to the summing of the local outputs,

[iii] Kuipers/Aström use the notion 'control law' in the sense of equations rather than IF - THEN rules!

this approach is likewise limited to static equations for the local control laws (see section 3.3). The Takagi-Sugeno model is, however, more compact and computationally efficient than the approach suggested by Kuipers and Aström.

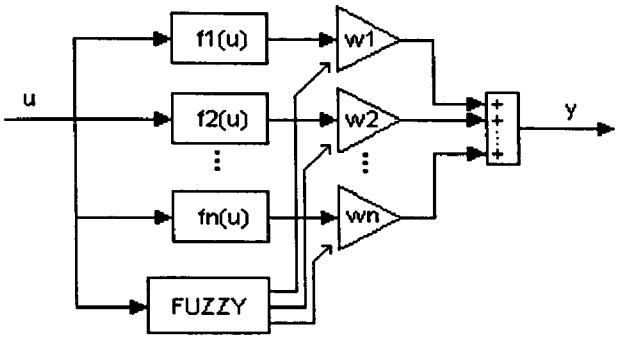


Fig. 3: The Heterogeneous Control System Approach

Another widely applied approach that combines the advantages of both conventional control engineering and fuzzy techniques is the adaptation of PID controllers via fuzzy adapters [23] (see Fig. 4). Like the 'heterogeneous control law' approach, this concept has specifically been applied to the design and implementation of fuzzy hybrid control systems. The success of this approach in many industrial applications [24] is largely based on the wealth of conventional analysis and validation techniques which are important to guarantee the stability of critical processes. This close relation to traditional techniques also increases the acceptance of such hybrid systems in industry.

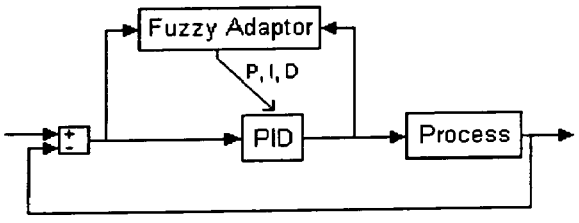


Fig. 4: Fuzzy Adapter for PID controller

3. Suggestion Of A New Fuzzy Hybrid Approach To Process Modelling

The proposal is closely related to the Takagi-Sugeno model in that it is likewise aimed at expressing very nonlinear functional relations in a simplified, efficient manner using piecewise linear equations that are combined via the fuzzy notion. Both input and output of this model are therefore normally also 'crisp', although an extension to express parameter uncertainty similarly to Grobbelaar [17] would be possible.

To make the fuzzy-hybrid approach applicable for dynamic system modelling and simulation, the major aim of the proposed modelling concept is to overcome the above mentioned limitations with respect to the integration of any kind of dynamic system equations. In spite of the close relation to the Takagi-Sugeno model in terms of its aim and functionality, this process modelling approach is actually derived from the fuzzy adaptation of PID controllers as Fig. 5 shows.

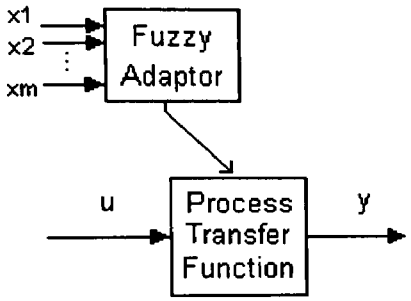


Fig. 5: The Suggested Fuzzy Hybrid Approach to Process Modelling

Instead of varying the P, I and D parameters of a controller using a fuzzy adapter, the application as a compound simulation model of the process itself requires the consideration of any polynomial or physically meaningful parameter, like time constants and damping ratios. Through the adaptation of these latter parameters, it is now possible to simulate truly nonlinear dynamic processes which is not the case with any of the approaches discussed in the previous section. Rather than calculating local system outputs and averaging these as in the Takagi-Sugeno or heterogeneous control law approaches, the average parameters are first aggregated according to the current operating condition and transferred to the single overall system equation from which the output of the model is determined. This system equation may be of any kind - especially dynamic (for example a transfer function or differential equation).

3.1 *Why* This Fuzzy Hybrid Approach?

As discussed in the introduction, the reason for developing a new modelling approach is the lack of an appropriate approach that addresses the need of simplified nonlinear multivariable modelling for practitioners in industry.

The application of a fuzzy hybrid concept for this purpose is particularly sensible, because

- it enables the use of locally valid, linear system equations which are the most probable format of useful process information that is either available or that can be derived from process experience in industry [4];

- it is an efficient way of combining these locally valid equations to a dynamically adapted nonlinear multivariable model;
- conventional analysis techniques (e.g. stability) can still largely be applied;
- the fuzzy parameter interpolation, together with the simplifications that are detailed in the following section, is more versatile than conventional interpolation approaches.

The latter aspect refers in particular to the fact that, unlike the conventional approaches, the fuzzy interpolation covers also multiple valued parameter variations like hystereses or even n-dimensional, multiple variable relationships within the same concept, if the appropriate influences are specified[iv].

The advantages of the overall approach are discussed in section 3.5 of this paper.

3.2 The Modelling Sequence Using The Suggested Fuzzy-Hybrid Approach

This section details the above stated general idea by introducing a set of modelling steps together with some important selections and defaults that make the approach very efficient and enable its automation while retaining its flexibility. Using these defaults, the user of such an automated approach would not be required to understand the details of fuzzy modelling. For the introduction of the stepwise procedure, a hypothetical example process without a particular physical manifestation was chosen, which is both simple and illustrative, showing the particular characteristics of the approach.

The parameters of the example process vary, depending on the influence parameter 'INFLUENCE'. In addition, the structure changes from 2nd order proportional behaviour at low levels of 'INFLUENCE' to 1st order at high levels of 'INFLUENCE'.

The modelling sequence - steps 1) to 8):

1) *Determination of the influences on the nonlinear behaviour of the process.* These influences which are responsible for the transitions between different characteristics (i.e., system equations) of the process are the inputs to the fuzzy adapter. The preselection of input candidates and the determination of input variables form the 'structure identification I' according to Sugeno and Yasukawa [8]. In the case of multiple influences, only the one that is expected to be most important should initially be considered. If the model quality proves to be insufficient, further influences can be added to the model in a stepwise fashion so that the previous version of the model can always be re-used.

[iv] please refer to the example of multiple variable modelling in section 4.

example: The parameter 'INFLUENCE' of the example process causes the changes in the process characteristic. With 'INFLUENCE', the fuzzy adapter will therefore have one input.

2) *Partition of the input space.* The question

"How many characteristic behaviours of the overall system can or should be distinguished and which are the related input conditions to the fuzzy adapter?" must be answered. To keep the modelling effort at a minimum, this number should initially be quite low (typically two to five). If required, the model quality can be increased at a later stage by considering further operating conditions.

example: Three characteristic levels of 'INFLUENCE' (5, 11 and 20) are distinguished.

3) *Collection of the local process equations.* The source of this information could either be any conventional process identification using a small perturbation approach or even a simple identification of the local process behaviour on the basis of experience or simple step response tests together with look-up tables (e.g. Strejc's method [25]). Generally, however, the local equations should be kept as simple as possible.

example:

The following three typical transfer functions are assumed to have been found through a small perturbation approach:

OP1: at INFLUENCE = *5* $$G_5(s) = \frac{10}{9s^2 + 1.2s + 1}$$

OP2: at INFLUENCE = *11* $$G_{11}(s) = \frac{6}{25s^2 + 4s + 1}$$

OP3: at INFLUENCE = *20* $$G_{20}(s) = \frac{15}{7s + 1}$$

("OP" stands for operating point of the nonlinear influence parameter, here named 'INFLUENCE')

4) *The different local process equations with fixed parameters are combined to a single system equation with variable parameters.* If the local process equations are of different order, the highest order equation is chosen.

example: $G_{example}(s) = y / u = b_0 / (a_2 s^2 + a_1 s + 1)$

In applications where the global **absolute** values are required, the computation of the static gain characteristic in a separate equation becomes necessary (cf. section 3.4). An example is given in section 4.

5) *The membership functions for the input variables to the fuzzy adapter are generated.* For reasons of simplicity, the default triangular membership functions, which are easily generated around the known operating points of the input variables, should be used. Also, the default functions fulfil the constraint $\Sigma\mu_i = 1$ for any input in the operating range. With these simplifications, the fuzzified inputs contain already the complete information for the weighted aggregation of the output values of the fuzzy adapter.
Fuzzified values are written in a row vector: ms = [μ_{OP1}, μ_{OP2}, ..., μ_{OPn}].

example: membership functions around the operating points 5, 11 and 20

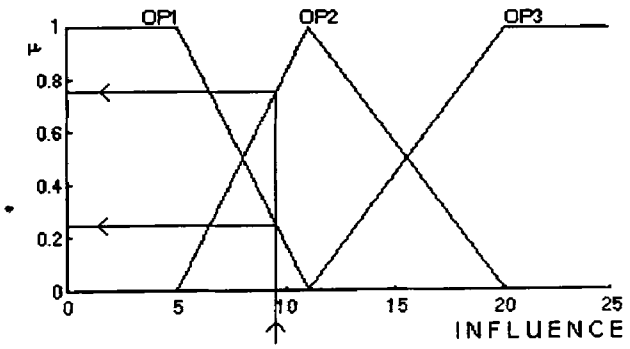


Fig. 6: Membership Function, Example

The fuzzified input value 9.5 is written as the row vector
ms(9.5) = [0.25, 0.75, 0] in this example.

6) *The rule-bases for the different parameters are specified.*
Without limiting its general applicability, the Takagi-Sugeno model gains part of its efficiency from assuming singletons as output membership functions and fixing the implication and aggregation methods. For the suggested fuzzy-hybrid modelling approach, the same conventions are defined.
Using singletons as output sets, the rule bases can be summarised to very simple matrices. The single and dual variant cases are particularly easy to handle, yielding column vectors or two dimensional matrices, respectively, as rule bases. Each parameter of the system equation requires a separate rule-base matrix which is simply an ordered collection of the crisp values that the particular parameter takes on for the known operating conditions. The matrices are therefore directly taken from the equations in the third step of this sequence. The order of the matrix elements must be consistent with the membership row vectors - in a two-input case with the columns of the rule matrices reflecting the

parameter changes due to the first input ('*A*') to the fuzzy adapter and the rows reflecting the changes due to the second input ('*B*') and increasing matrix element indices referring generally to increasing absolute operating point levels of the inputs to the fuzzy adapter. The structure of the rule-base matrix for any parameter "*xy*" which is dependant on the values of the process parameters *A* and *B* is:

$$RBxy = \begin{bmatrix} r_{A1B1} & r_{A1B2} & \cdots & r_{A1Bm} \\ r_{A2B1} & r_{A2B2} & \cdots & r_{A2Bm} \\ \vdots & \vdots & & \vdots \\ r_{AnB1} & r_{AnB2} & \cdots & r_{AnBm} \end{bmatrix}$$

This rule-base matrix is the short format of the rule-base

IF $A == A1$ AND $B == B1$ THEN $xy = r_{A1B1}$
IF $A == A2$ AND $B == B1$ THEN $xy = r_{A2B1}$

•
•
•

IF $A == An$ AND $B == Bm$ THEN $xy = r_{AnBm}$

Each element r_{AiBj} of the rule-base matrix represents therefore a rule. The rules have all the same weight (= 1) and the antecedents of a rule can only be combined by 'AND'. Thus, different antecedents that yield the same result (and could normally be combined by 'OR') must be put into separate rules.

example: In the simple, single variant case considered here as an example, the rule-bases are merely 3x1 - column vectors with their elements (i.e., parameter values) ordered according to the operating points.

$$RBb_0 = \begin{bmatrix} 10 \\ 6 \\ 15 \end{bmatrix}, \qquad RBa_1 = \begin{bmatrix} 1.2 \\ 4 \\ 7 \end{bmatrix}, \qquad RBa_2 = \begin{bmatrix} 9 \\ 25 \\ 0 \end{bmatrix}.$$

7) *Fuzzy implication, aggregation and defuzzification.* These steps take full advantage of the simplifications suggested in steps 5) and 6) in that they can be combined into a single, simple and computationally efficient matrix operation: using multiplication as implication method and aggregating simply the singletons, the multiplication of the membership vectors ("*ms..*") with the rule-base matrices ("*RB..*") combines all three steps. For a fuzzy

adapter with two inputs, '*A*' and '*B*', this multiplication is carried out for all *k* parameters of the system equation as follows:

$$ms_A(x_A 0) \times RB_k \times ms_B(x_B 0)^T =$$

$$\begin{bmatrix}\mu_{A1} & \mu_{A2} \cdots\cdots & \mu_{An}\end{bmatrix} \times \begin{bmatrix} r_{A1B1} & r_{A1B2} & \cdots & r_{A1Bm} \\ r_{A2B1} & r_{A2B2} & \cdots & r_{A2Bm} \\ \vdots & \vdots & & \vdots \\ r_{AnB1} & r_{AnB2} & \cdots & r_{AnBm} \end{bmatrix}_k \times \begin{bmatrix}\mu_{B1} \\ \mu_{B2} \\ \vdots \\ \mu_{Bm}\end{bmatrix}$$

(the row vector $ms_B(x_B 0)$ must be transposed, (^{T});
$x_A 0$, $x_B 0$ are any input values to the fuzzy adapter)

example: The input value 9.5 is fuzzified to ms(9.5) = [0.25, 0.75, 0]; the transfer function parameters for this operating condition are calculated by multiplying the membership row vector ms(9.5) with the rule-base matrices.

$$b_0(9.5) = ms(9.5) \times RBb_0 = \begin{bmatrix}0.25 & 0.75 & 0\end{bmatrix} \times \begin{bmatrix}10 \\ 6 \\ 15\end{bmatrix} = \underline{\underline{7.00}}$$

$$a_1(9.5) = ms(9.5) \times RBa_1 = \begin{bmatrix}0.25 & 0.75 & 0\end{bmatrix} \times \begin{bmatrix}1.2 \\ 4 \\ 7\end{bmatrix} = \underline{\underline{3.30}}$$

$$a_2(9.5) = ms(9.5) \times RBa_2 = \begin{bmatrix}0.25 & 0.75 & 0\end{bmatrix} \times \begin{bmatrix}9 \\ 25 \\ 0\end{bmatrix} = \underline{\underline{21.00}}$$

8) *Transfer of the new parameter values to the linear parametric system equation and evaluation of the updated equation.* For an incrementally continuous simulation, this step, together with the matrix operation of step 7 and the fuzzification of the input signals can easily be programmed in any simulation environment and evaluated at fixed time increments.

example:

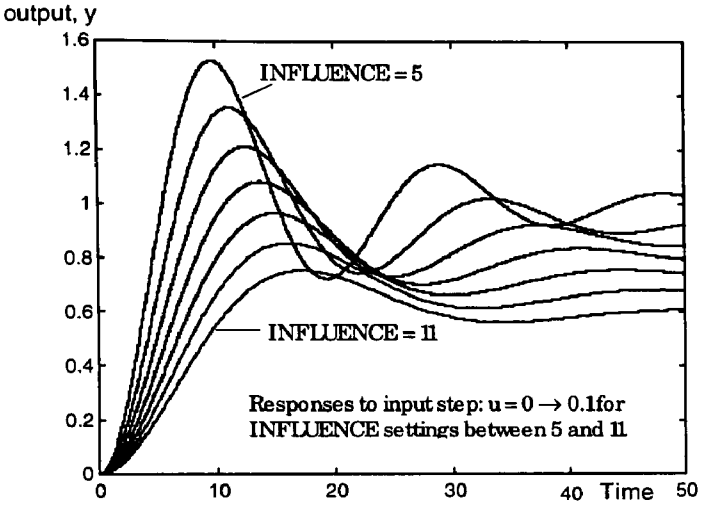


Fig. 7: Step Responses of the Fuzzy Adapted Transfer Function at Different Settings of 'INFLUENCE' Between the Initially Known Levels 5 and 11.

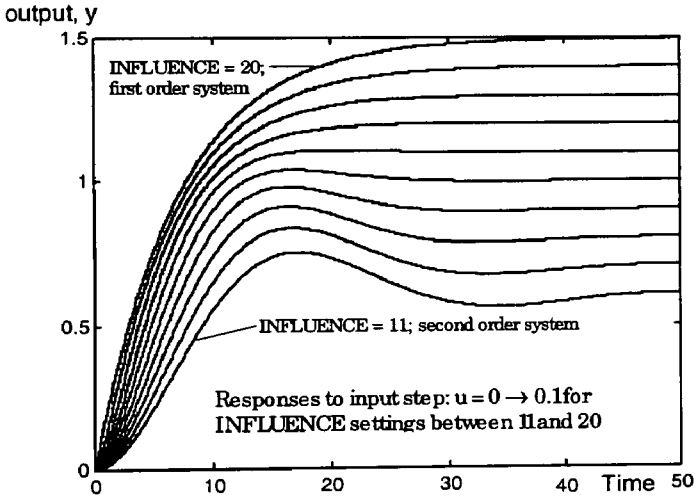


Fig. 8: Step Responses of the Fuzzy Adapted Transfer Function at Different Settings of 'INFLUENCE' Between the Initially Known Levels 11 and 20.

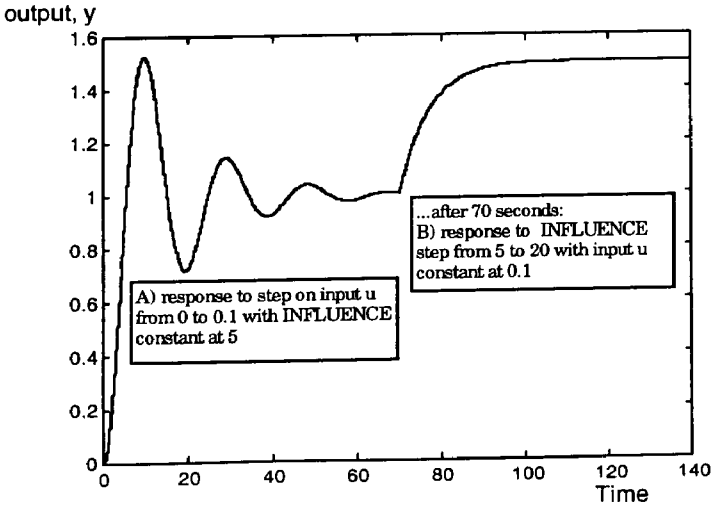


Fig. 9: Successive Responses of the Fuzzy Adapted Transfer Function to a Step on Input 'u' and on the Parameter 'INFLUENCE'.

The simulation results that are shown in Fig. 7, Fig. 8 and Fig. 9 illustrate some of the important properties of the suggested approach:

- a continuous frequency shift in the oscillations of the second order responses between the initially specified parameter levels of INFLUENCE = 5 and 11
- the transition from second to first order behaviour in Fig. 8
- the multivariability, with dynamic responses to changes of both the input, 'u', and the influence parameter ('INFLUENCE') in Fig. 9.

Using the introduced simplifications as default settings for an automated approach to fuzzy hybrid modelling, all the user needs to supply is a set of operating points with associated locally valid SISO system equations. The surprisingly simple nonlinear multivariable modelling approach for practitioners is therefore based on both the suggested fuzzy hybrid modelling as such and the default settings with respect to input and output membership functions, implication, aggregation and defuzzification methods and the simple matrix notation.

3.3 Limitation of the Takagi-Sugeno approach

Being based on locally valid system equations and using a fuzzy hybrid concept, the nonlinear process modelling approach by Takagi and Sugeno [2] is particularly closely related to the approach that is proposed in this paper. Although structurally not quite as elegant and computationally not as efficient, the concept of the heterogeneous control system approach described by Kuipers and Aström [22] is conceptually the same as Takagi-Sugeno's.

As discussed above, both the Takagi-Sugeno approach and Kuipers-Aström's are based on averaging the output values of the locally valid system equations to obtain global results. While this kind of interpolation is a correct approximation for static relationships, averaging dynamic trajectories is theoretically improper. Although under certain circumstances the weighted interpolation of trajectories can yield good results, the limitation becomes particularly apparent if oscillating signals are considered: In the case that, for example, the system response to an oscillating input signal is a sinusoid signal whose phase angle depends on another influence parameter, the local trajectories cancel each other to some extent, possibly even fully (180° phase difference), when they are averaged to the global output.

In Fig. 10, an intermediate system response of the example from section 3.2 is determined according to Takagi-Sugeno (or equivalently to Kuipers-Aström) and shown together with the results of the approach introduced in this paper. While the Takagi-Sugeno model follows in terms of frequency and damping only the stronger oscillations of the trajectory for the pre-defined local model at INFLUENCE = 5, the suggested model shows correctly an intermediate gain, damping and frequency of the trajectory (see also Fig. 7).

Since oscillations cannot be avoided in most dynamic simulations and also because the validity of process models should not depend on the type of input signal, the Takagi-Sugeno and Kuipers-Aström models should be built exclusively from static equations, which relate only instantaneous process variables.

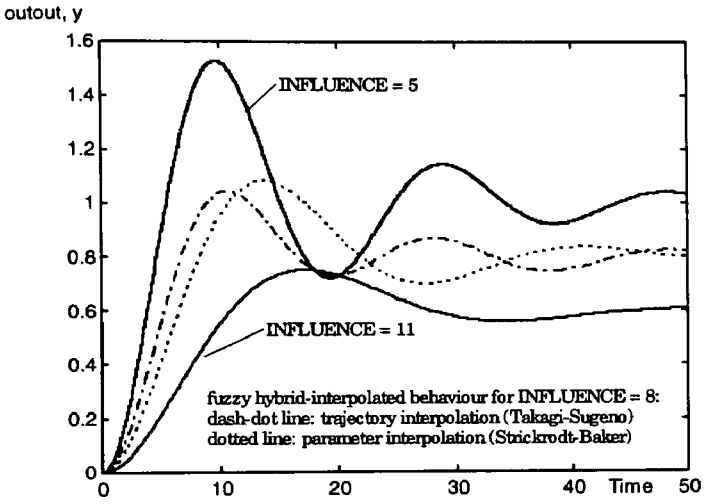


Fig. 10: Step Response at Intermediate Level according to Takagi-Sugeno and Strickrodt-Baker

3.4 The Standard Structure Of The Global Fuzzy Hybrid Model

Global fuzzy hybrid process models consist, apart from the fuzzy adapter, of two modules (Fig. 11): a block with the dynamic part of the parametric, fuzzy adapted transfer function ('fuzTF') and a preceding fuzzy static characteristic ('fuzSC'). The parametric fuzSC-function has the standard equation structure of a straight line:

$$Y = Kp \cdot U + Yo \qquad \text{or} \qquad Y = b_0 \cdot U + Yo$$

with

U = main input to the overall model; global value
Y = global steady state output
Kp = proportional gain in 'physically meaningful' terminology
b_0 = proportional gain in polynomial terminology
Yo = offset of the straight line

For the specification of the rule base RBYo, the offset Yo is calculated as

$$Yo = Y - Kp \cdot U \qquad \text{or} \qquad Yo = Y - b_0 \cdot U$$

from the known operating points (U,Y) and the gain Kp (or b_0) of the associated local transfer function.

The tangency of the straight line to the n-dimensional surface (n = number of influences fed into the fuzzy adapter + 1) of the static characteristic is achieved by the parameter adaptation through the fuzzy module. For each operating point, the 'fuzSC' represents therefore the locally valid, linearised static relationship and is thus the static equivalent of the dynamic 'fuzTF' relationship.

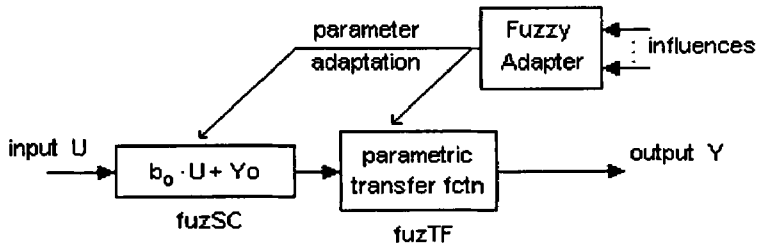


Fig. 11: Standard Structure of the Global Fuzzy Hybrid Model

Using the 'Hammerstein-like' structure of the fuzzy static characteristic preceding the dynamic block as the ***standard structure of the global fuzzy hybrid model***, it is possible to accommodate different conventional model structures. The static characteristic of the considered funnel tank in section 4, for example, would have to be modelled conventionally by a 'Wiener' structure with the static characteristic following the dynamic block. The fuzzy hybrid model, however, can represent the real process behaviour in its standard structure simply by using the overall output as an influence parameter to the fuzzy adapter. A conventional 'Hammerstein' structure would be modelled in fuzzy hybrid terms by applying the overall input as an influence parameter for the fuzzy adaptation. Additionally, the standard fuzzy hybrid structure accommodates far more complex structures than 'Hammerstein' or 'Wiener', if any influence parameter - or even several of them - other than the overall input or output is used.

3.5 The Advantages Of The Suggested Fuzzy-Hybrid Modelling Approach

This section summarises some of the advantages of the proposal compared with existing modelling approaches - in particular those that are based on partial modelling information.

Most notably, the approach as such, the modelling sequence and the resulting model are exceptionally simple:

- The model structure is easy to understand - even with very little fuzzy logic knowledge.
- The fuzzy hybrid model has a standard block structure: for global process models, the fuzzy static characteristic *always* precedes the fuzzy adapted dynamic block

(Hammerstein-style), even if the process would have to be modelled conventionally using a Wiener- or other, more complex structures.

- The model can be easily programmed in a general purpose simulation environment.
- Since it takes advantage of the suggested simplified standard membership functions and implication, aggregation and defuzzification methods as default settings, the modelling approach can be largely automated, hiding in particular all parts of the fuzzy approach, if required. The defaults should only be edited by more experienced modellers.
- Once automated, the effort for building nontrivial models is extraordinary small.
- The only required modelling information is a set of operating points with associated locally valid SISO system equations which can be easily acquired using simple identification approaches (cf. [4]).

This simplicity of the approach is well in line with the general idea behind modelling: since the model-behaviour is always only an approximation of the real behaviour, the model must be kept as simple as possible for a specific application [26]. The suggested approach follows therefore this philosophy as an extension of simple linear dynamic modelling towards modelling a class of nonlinear dynamic processes.

Despite its simplicity, however, the modelling approach is powerful in that it can combine the local dynamic process knowledge to highly nonlinear, multivariable global models. The particular benefit for practical users is therefore the availability of an approach that enables the modelling of such complex processes despite the various constraints in industry that have been discussed in the introduction.

The quality of a model of the suggested type is mainly dependent on the quality of the local system equations as well as the partition of the input space. The approach as such is therefore as applicable to coarse modelling on the basis of assumptions, separating, for example, only two characteristic operating conditions, as it is to the combination of several well determined local system equations for precise predictions of the process behaviour.

Further, it should be noted, that the suggested approach is in principle not limited to the combination of piecewise linear dynamic system equations but could likewise integrate some types of nonlinearities that change under different process conditions. An example of particular practical importance is varying dead-times. Although our current investigations are focused on piecewise linear systems, an extension towards some nonlinearities is merely an implementational question.

Another major advantage of the suggested modelling approach is its modularity that helps to keep the model complexity and modelling effort to a minimum: after considering initially only the most important influence on the nonlinear behaviour of the process, further influences can

be added to the model at a later stage if required. Likewise, the model quality can be improved in a stepwise manner by adding intermediate operating conditions to the rule bases and membership functions. Previous versions of the process model can always be re-used in such situations.

The unique advantages of the suggested fuzzy hybrid approach as well as the combination of these characteristics make it particularly useful for the modelling of dynamic large scale systems in industry.

4. An Application Example

To illustrate the above listed advantages of the suggested modelling approach, this section gives an application example for a real process. A water tank in the shape of a funnel (Fig. 12) is considered here. This tank behaves very nonlinearly with respect to its gain and time constant due to the outflow equation (square root function) and the variation of the tank area over its height. The chosen main process input is the water flow into the tank, Q_{in} [m^3/s] and the output is the water level, h [m]. Additionally, the varying outlet area A [m^2], which is operated by a valve, is considered as a secondary process input.

The funnel tank is a good demonstration example for the application of the modelling approach in process industry:

- It behaves very nonlinearly.
- It is multivariable. Initially, the process could be modelled with constant outlet area and later it could be extended to cover varying outlet areas (stepwise model extension). In this presentation, however, the complete process is modelled in one step.
- The process is stable and therefore enables the use of very basic open-loop identification approaches.
- Typical for process industry, the time constants are big enough to allow for the application of an 'intelligent' modelling interface that aims at translating the user's process experience into a model.

As opposed to the funnel tank, unstable and fast test processes like the inverted pendulum are not appropriate systems to show the validity of this nonlinear multivariable modelling approach, because they do not address its particular purpose. This is not to say that the introduced model could not handle such processes. However, it is important to stress again that the goal and uniqueness about the proposed model is not the time-critical performance in simulation experiments, but the ease with which a very satisfactory, highly nonlinear, multivariable process model can be derived from quite basic information that can quickly be

obtained for stable processes - even in an industrial environment and by engineers that are no modelling experts.

A direct comparison with other modelling approaches that are based on partial process information is not shown here, because these are either based on very different assumptions with respect to the available modelling information, or are - like the Takagi-Sugeno or Kuipers-Aström approaches - inapplicable to processes with varying dynamic characteristics (e.g. time constants, damping ratios), as discussed and illustrated in section 3.3. However, the simulation results of the fuzzy hybrid model are compared with the theoretically derived process model, which represents in this case anyway the 'ideal' behaviour.

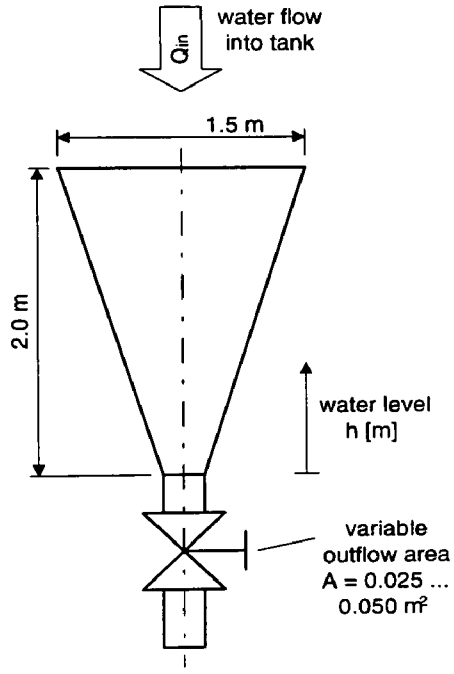


Fig. 12: The Funnel Tank

The modelling sequence:

1) In this particular example, the system **output** (water level) influences the dynamic behaviour of the real process and is therefore also in the model fed back as the **input** to the fuzzy module which adapts the parameters of the transfer function. Thus, the water level, h, is the first input to the fuzzy adapter. It is important to note that this exceptional situation of an internal feedback in the model imposes a particular challenge to the quality of the global process model and its simulation results. The external influence 'outlet area' is an additional input to the fuzzy adapter.
2) a) Four different water levels were considered: $h_1 = 0.131$m, $h_2 = 0.566$m, $h_3 = 1.094$m and $h_4 = 1.616$m, which are the steady state levels that resulted from the small perturbations of the process to determine the local transfer functions.
   b) Three outlet areas are distinguished: $0.025m^2$, $0.035m^2$ and $0.050m^2$.

3) From the local step responses (first order proportional behaviour), the characteristic gains and time constants were determined as follows (Table I), using the simple tangent evaluation at t = 0 :

Table I: Local Transfer Functions

	$A_1 = 0.025m^2$	$A_2 = 0.035m^2$	$A_3 = 0.050m^2$
$h_1 = 0.131m$	$G_{11}(s) = \frac{5.7}{0.39s+1}$	$G_{12}(s) = \frac{4.3}{0.36s+1}$	$G_{13}(s) = \frac{3.1}{0.32s+1}$
$h_2 = 0.566m$	$G_{21}(s) = \frac{12.8}{2.62s+1}$	$G_{22}(s) = \frac{9.3}{2.29s+1}$	$G_{23}(s) = \frac{6.6}{1.88s+1}$
$h_3 = 1.094m$	$G_{31}(s) = \frac{18.1}{8.47s+1}$	$G_{32}(s) = \frac{13.1}{7.08s+1}$	$G_{33}(s) = \frac{9.4}{6.34s+1}$
$h_4 = 1.616m$	$G_{41}(s) = \frac{22.1}{20.14s+1}$	$G_{42}(s) = \frac{16.0}{16.28s+1}$	$G_{43}(s) = \frac{11.6}{12.87s+1}$

4) The parametric system equation which is valid throughout the operating range is therefore:

$$G(s) = \frac{K_p}{T \cdot s + 1}$$

Using this structure, the resulting fuzzy hybrid model can predict very well any local changes to any intermediate level within the operating range. It is therefore applicable to many closed loop simulations. In this example, however, the aim is to build a global model that predicts the output, or water level, within the operating range in absolute terms on the basis of the absolute input signal (Q_{in}). For this purpose, the adapted gain Kp is considered together with a likewise adapted offset Yo (Table II) of the water level in a separate static equation according to the standard structure of the global fuzzy hybrid model (Fig. 13).

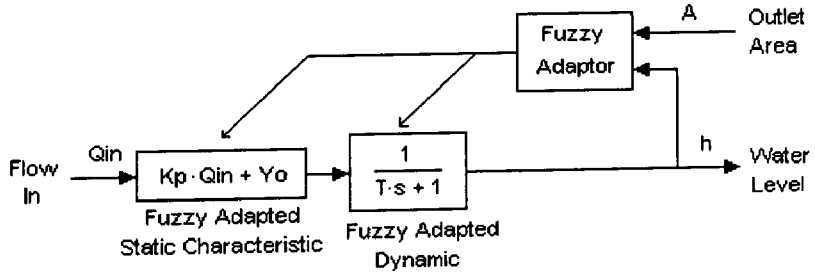


Fig. 13: Fuzzy Hybrid Model of the Funnel Tank with Variable Outlet Area

All required process parameters are summarised in the following Table II.

Table II: Summary of the process parameters

A [m²]	h [m]	Q_{in} [m³/s]	Kp	Yo *)	T
0.025	0.131	0.0400	5.7	-0.098	0.39
	0.566	0.0833	12.8	-0.499	2.62
	1.094	0.1158	18.1	-0.999	8.47
	1.616	0.1407	22.1	-1.499	20.14
0.035	0.131	0.0560	4.3	-0.107	0.36
	0.566	0.1166	9.3	-0.518	2.29
	1.094	0.1621	13.1	-1.028	7.08
	1.616	0.1970	16.0	-1.534	16.28
0.050	0.131	0.0800	3.1	-0.114	0.32
	0.566	0.1665	6.6	-0.533	1.88
	1.094	0.2315	9.4	-1.076	6.34
	1.616	0.2814	11.6	-1.646	12.87

*) $Yo = h - Kp\ Q_{in}$

5) The default triangular membership functions with $\Sigma\mu_i = 1$ are applied here for both influences. The peaks of the fuzzy sets ($\mu = 1$) are set at the distinguished water levels and outlet areas given in 2):

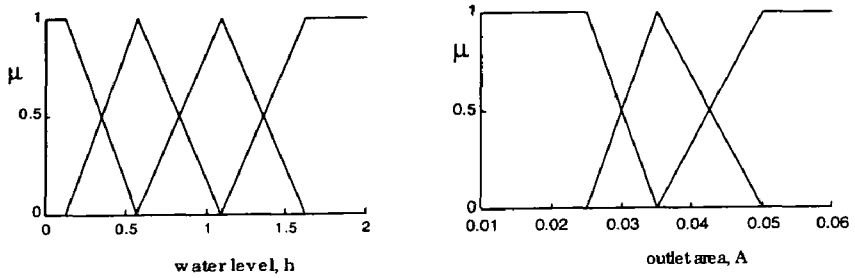


Fig. 14: Membership Functions for the Funnel Tank Model

6) The rule bases are defined as follows:

$$RBKp = \begin{bmatrix} 5.7 & 4.3 & 3.1 \\ 12.8 & 9.3 & 6.6 \\ 18.1 & 13.1 & 9.4 \\ 22.1 & 16.0 & 11.6 \end{bmatrix} \qquad RBYo = \begin{bmatrix} -0.098 & -0.107 & -0.114 \\ -0.499 & -0.518 & -0.533 \\ -0.999 & -1.028 & -1.076 \\ -1.499 & -1.534 & -1.646 \end{bmatrix}$$

$$RBT = \begin{bmatrix} 0.39 & 0.36 & 0.32 \\ 2.62 & 2.29 & 1.88 \\ 8.47 & 7.08 & 6.34 \\ 20.14 & 16.28 & 12.87 \end{bmatrix}$$

7) As described in section 3, the fuzzy implication, aggregation and defuzzification were simply summarised by multiplying the fuzzified inputs to the fuzzy adapter (i.e., the membership vectors of the water level and the outlet area) at each simulation increment with the rule base matrices, yielding updated parameter settings for Kp, T and Yo.
8) The updated parameters are transferred to the static and dynamic system equations which are evaluated in the new continuous simulation increment.

Using the physical balance equations, a process model of the funnel tank was derived theoretically. This model was simulated concurrently with the fuzzy hybrid model in the validation tests and served as a direct reference.

The signals for the simulation (cf. Fig. 15, Fig. 16 and Fig. 17) were defined in particular to explore the model behaviour in intermediate areas between the previously known points. The particular shapes of the signals in Fig. 15 and Fig. 16 are quite meaningless for the real funnel tank but serve the purpose of investigating how well the fuzzy hybrid model (dotted line) handles rather more 'awkward' situations in comparison with the ideal behaviour (full line).

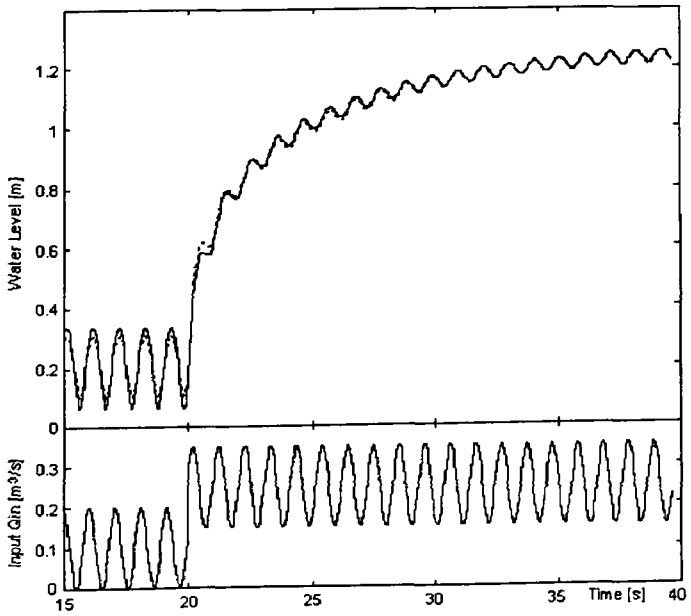


Fig. 15: Simulation Results of the Fuzzy Hybrid (Dotted Line) and the Theoretically Derived Funnel Model (Full Line) at A = 0.05 m²

The sinusoid input signal with constant frequency and amplitude but changing offset (Fig. 15) is a very severe test to the fuzzy adapter - in particular during the transition from low to high water level. The fact that the fuzzy hybrid signal follows the ideal response so well with respect to gain and phase indicates the successful continuous adaptation of the *dynamic* properties.

It is, however, especially interesting to see in Fig. 16 the fuzzy model responding dynamically to the variations of the outlet area 'A', since information with respect to the dynamic relationship h = f(A) was not directly given in the modelling process: the dynamic process information was limited to the main $h = f(Q_{in})$ relations at certain static settings of the influence parameters. The positive simulation results illustrate therefore how a simple parametric transfer function is 'promoted' to a multivariable dynamic process model by tuning its parameters according to the influences.

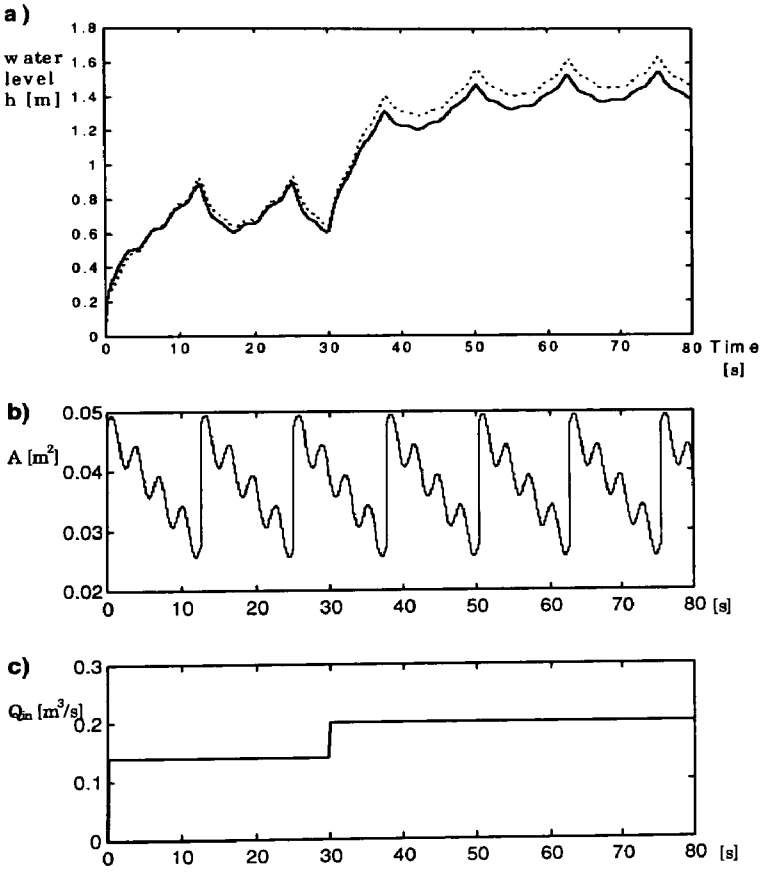


Fig. 16: a) - Simulation Results of the Fuzzy Hybrid (Dotted Line) and the Theoretically Derived Funnel Model (Full Line); b) and c) - The related input signals that cause the above response a)

Although the water level is, like the outlet area, an influence to the dynamic process behaviour, it is, as opposed to A, not an additional input to the overall model, but an internal influence that is fed back. It has therefore more the characteristic of a dynamic nonlinear influence.

Bearing in mind that the global (nonlinear and multivariable) fuzzy hybrid model was only derived from local, linear SISO (single input - single output) process information, the model performs very well in the direct comparison with the 'original' process behaviour. Apart from the relatively little process information and the challenging test signals, the internal feedback of the model output as an influence imposes a particular difficulty: The output of any model is

always an approximation of the real output - in the funnel tank example, the output is therefore fed back together with its model error and re-used for the calculation of updated parameters. This bears the risk of an accumulating prediction error. Wherever possible, such internal feedbacks should therefore be avoided.

Inevitably, the consideration of an increasing number of input (or influence) variables in the process model results also in the combination of uncertainties and interpolation errors, as the comparison of Fig. 16 ($\mathbf{Q_{in}}$, **A** and **h** varying at the same time) with Fig. 15 and Fig. 17 (A = constant) shows. Nevertheless, the application of multivariable process models is normally of great advantage to the simple neglection (i.e., linearisation) of influences, even if the available process information is quite basic, as the example shows.

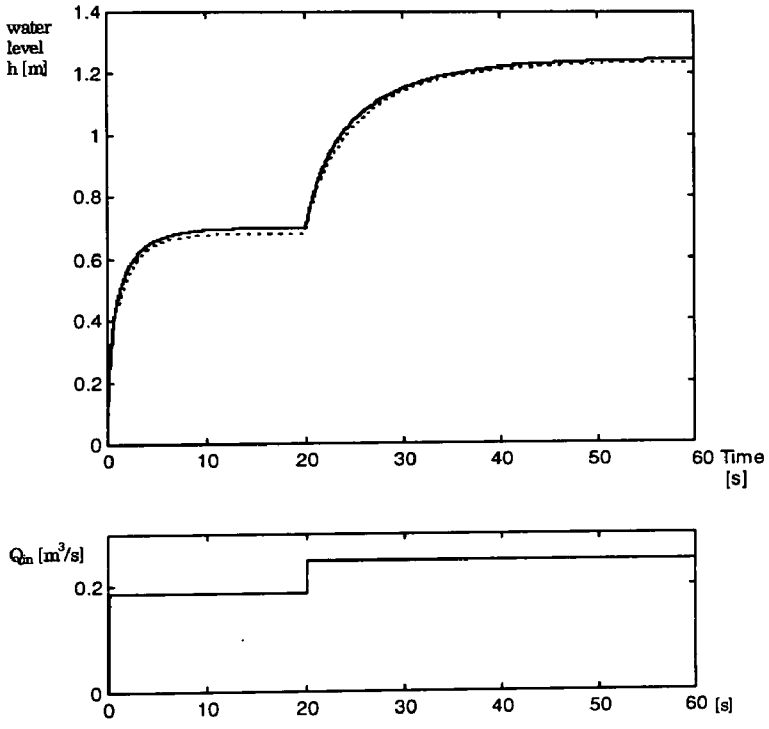


Fig. 17: Global Step Responses at the two Intermediate (i.e. previously not defined) Steady State Levels h = 0.7m and 1.25m (A = 0.05 m²)

Overall, the fuzzy hybrid model performs very well, without the tendency of an increasing output error and representing even the phase and amplitude changes during transition periods as well as the effects of concurrent variations of all three inputs/influences (Q_{in}, A and h) correctly.

5. Conclusion

Like the Takagi-Sugeno model, the suggested approach is suited to process identification on the basis of measured data. In fact, the results of the structure and parameter identification steps suggested by Takagi-Sugeno [2] and Sugeno-Kang [9] could also be used with the fuzzy-hybrid model which has been introduced in this paper. Rather than trying to identify the fuzzy hybrid model directly from process data, the main aim of this approach, however, is the integration of process information in the form of locally valid, basic system equations that can easily be obtained even in an industrial environment, using either an easy-to-use linear identification tool or a knowledge acquisition approach that translates process experience into linear transfer functions [4].

Both this and the Takagi-Sugeno approach benefit from simplifications that make the algorithms more efficient; the simplifying assumptions of both approaches are largely the same (e.g. singletons as output membership functions, multiplication as implication method).

The Takagi-Sugeno approach further benefits from the low number of rules required and from the fact that the linear equations form an integral part of the fuzzy model, while the introduced approach necessitates in the same modelling situation the number of varying parameters times the number of rules of Takagi-Sugeno's model. However, the rule-bases in the proposed approach are merely simple matrices which are directly read from the local equations and the implication, aggregation and defuzzification are extremely efficient matrix operations. Instead of evaluating a separate system equation for every rule, only a single equation with the adapted parameters is evaluated. Nevertheless, the more 'integrated' characteristic of Takagi-Sugeno's model which does away with the actual 'adaptation' of a somewhat separate component is still advantageous for the combination of piecewise linear *static* equations. Still though, when it comes to combining locally valid *dynamic* equations to an overall model, there is no way around the calculation of a single overall output trajectory in the general case. Therefore, the approach suggested in this paper appears to be a sensible extension of fuzzy hybrid modelling towards dynamic system modelling and thus a complement to Takagi-Sugeno's model for static relationships.

The suggested model will form part of an integrated process modelling approach that aims at making efficient use of different kinds of process knowledge [4, 5], simplifying and automating the procedures required to obtain different types of process models or at least structural process information. Integrating the fuzzy hybrid modelling approach into this intelligent modelling interface will enable a new way of simple and fast nonlinear dynamic and multivariable modelling on the basis of experience which is particularly geared at industrial requirements. As part of a collaborative research project, the process information acquired

through this approach will complement the improved identification experiment design, process simulation and controller design in separate modules concurrently being developed.

REFERENCES

[1] S. A. Billings, " Identification of nonlinear systems - a survey", *IEE Proceedings D*, vol. 127, no. 6, pp. 272-285, 1980.

[2] T. Takagi and M. Sugeno, " Fuzzy identification of systems and its applications to modeling and control", *IEEE Trans. on Syst., Man, and Cybernetics*, vol. SMC-15, no. 1, pp. 116-132, 1985.

[3] J. F. Pollard et al., " Process identification using neural networks", *Computers chem. Engng*, Pergamon Press, vol. 16, no. 4, pp. 253-270, 1992.

[4] M. Strickrodt, R. Schumann and K. Baker, " An integrated knowledge engineering approach to process modelling for CACSD", *Proceedings of IFAC'96 World Congress*, 1996.

[5] M. Strickrodt, R. Schumann, K. Baker, " Knowledge acquisition as part of a practical CACSD approach", *IEE Colloq. on Advances in CACSD*, Digest No: 96/061, pp. 8/1-8/4, 1996.

[6] A. Hammerstein, " Nichtlineare Integralgleichungen nebst Anwendungen", (in German), *Acta Mathematica*, vol. 54, pp. 117-176, 1930.

[7] S. A. Billings, S. Y. Fakhouri, " Identification of systems composed of linear dynamic and static nonlinear elements", *5th IFAC Symp. on Ident. and Syst. Par. Est.*, pp. 493-500, 1979.

[8] M. Sugeno and T. Yasukawa, " A fuzzy-logic-based approach to qualitative modeling", *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, Feb., 1993.

[9] M. Sugeno and G. T. Kang, " Fuzzy modelling and control of a multilayer incinerator", *Fuzzy Sets and Systems*, Elsevier Publishers B.V., vol. 18, pp. 329-345, 1986.

[10] M. Strickrodt, K. Baker, " Practical applicability of qualitative modelling approaches to process simulation: A survey", currently under review for publication in the *IEEE Trans. on Syst., Man, and Cybernetics.*

[11] J. Lunze, " Qualitative modelling of continuous-variable systems by means of non-deterministic automata", *Intell. Syst. Eng.*, vol. 1, pp. 22-30, 1992.

[12] J. Lunze, " Qualitative modelling of linear dynamical systems with quantized state measurements", *Automatica*, Pergamon, Elsevier Science Ltd, vol. 30, no. 3, pp. 417-431, 1994.

[13] A. Bredebusch, J. Lunze and H. Richter, " A Petri-Net representation of the qualitative behaviour of a dynamical continuous-time system", *Intelligent Systems Engineering*, IEE Conference Publ. No. 395, pp. 223-228, 1994.

[14] B. J. Kuipers, " Qualitative simulation", *Artificial Intelligence*, Elsevier Science Publishers B.V. (North-Holland), vol. 29, pp. 289-338, 1986.

[15] B. J. Kuipers, *Qualitative reasoning: modeling and simulation with incomplete knowledge*, MIT Press, 1994.

[16] R. Jain, " Analysis of fuzzy systems", *Fuzzy Automata and Decision Process*, pp. 251-268, 1977.

[17] G. B. Grobbelaar, " The use of fuzzy sets theory for modelling complex processes", *The Transactions of the SA Institute of Electrical Engineers*, vol. 81, no. 2, pp. 18-21, Jun., 1990.

[18] A. Kandel, " Imprecise modelling of systems and circuits in uncertain environments", *Proc. of the 1981 Europ. Conf. on Circuit Theory and Design*, Delft Univ. Press, pp. 984-987, 1981.

[19] R. Rouhani, " Bounding the behaviour of positive dynamic systems", *Journal of Economic Dynamics and Control*, vol. 2, pp. 283-299, 1980.

[20] D. Dubois and H. Prade, *Fuzzy sets and systems: theory and applications*, Academic Press, N.Y., 1980, chapter 2.

[21] H. Ushida, T. Yamaguchi, K. Goto and T. Takagi, " Fuzzy-neuro control using associative memories, and its applications", *Control Eng. Practice*, Pergamon Press Ltd, vol. 2, no. 1, pp. 129-145, 1994.

[22] B. Kuipers and K. Åström, " The composition and validation of heterogeneous control laws", *Automatica*, Pergamon, Elsevier Ltd., vol. 30, pp. 233-249, 1994.

[23] B.-M. Pfeiffer, " Selbsteinstellende klassische Regler mit Fuzzy-Logik", (in German), *Automatisierungstechnik*, vol. 42, no. 2, pp. 69-73, 1994.

[24] OMRON Corporation (Japan), *Fuzzy Guide Book*, 1992.

[25] R. Isermann, *Identifikation dynamischer Systeme 2*, (in German), Springer-Verlag, 1992, chapter 21.

[26] P. Eykhoff, *System identification*, John Wiley & Sons, 1974.

Practical Applicability Of Qualitative Modelling Approaches to Process Simulation: A Survey

for submission to the IEEE Transactions on Systems, Man, and Cybernetics

Matthias Strickrodt and Keith J. Baker[1]

Abstract – Qualitative modelling and simulation has become an extensive field of research which is difficult to review for someone who mainly wishes to apply a modelling technique that allows for incomplete system knowledge. This overview is aimed at engineers and scientists who are considering a qualitative modelling approach for a practical application. Firstly, attempts to clarify the role of qualitative approaches in the context of modelling in general are discussed and categories for grouping the reviewed approaches are defined. By putting the emphasis of this survey on a comparison of the main approaches with respect to the characteristics that are of particular importance in the domain of modelling and simulation in control engineering, the reader should get a feel for the applicability of the approaches in his/her specific field. Avoiding detailed descriptions of the algorithms involved, this paper gives a simplified introduction for the novice in the field of qualitative modelling and refers to the further literature.

1. Qualitative Process Modelling

Researchers from different fields - like process and control engineering, maths, computing, physics and artificial intelligence - have been working on 'qualitative modelling'. Due to the different backgrounds, misunderstandings and contradictory definitions are fairly commonplace. Therefore, Lunze [1] stated quite rightly in 1992 that there was not yet a clear and general definition of a qualitative model.

Quantitative or 'crisp' models (e.g. fully parameterised differential equations or transfer functions) are either not available or of little use if one of the following circumstances applies:

Manuscript received ...
[1] The authors are with the Department of Mechanical and Manufacturing Engineering, University of Glamorgan, Pontypridd Mid Glamorgan, CF37 1DL, U.K.
The research was carried out with the support of the British Council through the Anglo-German Academic Research Collaboration Programme ARC, Project 520, for the collaboration with the Fachhochschule Hanover.

- The dynamic system is not completely known with respect to the structure of the underlying differential equation or its parameter values.
- The actuator signals of complex plants are manually adjusted by an operator who takes various indicators of the current process state into account.
- Only coarse measured signals (e.g. high / medium / low) or indirect process state indicators are available.

Although this specification of the typical situations in which quantitative modelling is inapplicable and therefore qualitative or semi-quantitative modelling becomes a necessity is expressed in control engineering terms, it applies accordingly to all other strands of research. Based on the analysis of these typical situations, Lunze [2] suggested in 1993 his own definition: "Models, which are based on a coarse evaluation of signal and parameter values are denoted as 'qualitative models'. These models often refer to symbolic instead of numeric values with respect to the signals and their parameters."

As opposed to this open and general definition, many researchers (e.g. [3, 4]) consider 'qualitative modelling' as the generic term for all methodologies which use only the signs of parameters and influences.

In this survey, however, qualitative techniques in the latter sense are considered together with 'semi-quantitative' approaches that also use some sort of quantitative information. The view that any model which is not fully defined in a quantitative way is somewhat 'qualitative' - with varying degrees of quantitativeness - is therefore shared here (Fishwick [5]). While publications on control engineering applications of qualitative modelling often focus on the third of the above cited circumstances, the availability of only coarse measurements, this overview emphasises the first two aspects which refer to the incomplete knowledge of a process. Without the direct coupling to a process, exact input values can always be applied for simulation purposes.

The role of qualitative models in the general modelling context

To clarify the role of qualitative models within the general modelling context, it is useful to subdivide qualitative models according to their degree of quantitativeness. In section 3. of this overview, the different categories of qualitative models will be important for the discussion of the properties of the modelling and simulation approaches.

Different suggestions for such model categories have been made. Stevens et al. [7] categorised models hierarchically as follows:

- *Structural models* - containing only cause-effect and connectivity information
- *Qualitative models* - like structural models with additional qualitative dynamic information
- *Static models* - like structural models with additional numerical steady-state information
- *Quantitative models* - containing all the structural and numerical dynamic information about a system such that it is theoretically possible to obtain a complete and accurate description of the system's behaviour

These categories, however, are not truly hierarchical as the information in static models is no superset of the information covered by qualitative models. A more consistent suggestion was made by Mavrovouniotis and Stephanopoulos [8]:

- *Boolean models* represent only the existence of parameters and interrelations between them without any information on signs or magnitudes.
- Additionally, *qualitative models* represent the signs of variables and the direction in which each variable affects another one. Information on magnitudes or relative orders of magnitudes is not provided. A signed, directed graph (digraph) showing how parameters affect each other is an example of this type of model.
- *Order-of-magnitude models* provide, in addition to the information covered by qualitative models, some rough (absolute or relative) magnitudes of parameters and effects.
- *Quantitative models* employ the most detailed numerical and algebraic representations, such as systems of equations and numerical values of parameters.

This hierarchy is consistent but it is - like Stevens' model hierarchy - not complete for two reasons: firstly, because *heuristics*[2] based models, which play an important role in qualitative modelling, are not considered and secondly, because conventional quantitative models are not necessarily the best - or most detailed - abstraction of the real process. Mavrovouniotis himself corrected very recently [9] his old concept with respect to the most exact model and stated: "it is not crisp values that represent complete knowledge, since only one specific system in a specific state can be described by crisp values". He suggests lumping together many crisp descriptions in a "joint distribution function" for all the variables of a system as the most detailed form of knowledge.

It is, however, not possible to allocate *heuristics* based models appropriately within this model hierarchy since their degree of quantitativeness can vary significantly and they can cover information on the modelled process beyond the scope of quantitative models. A strictly hierarchical concept does therefore not appear to be an applicable format to order the different types of process models, unless the scope of the model ordering concept is explicitly restricted

[2] Heuristic knowledge is knowledge which points from problem features to problem solutions and can be formulated in if-then rules.

to *causal*[3] models. In this case, neither *heuristics* based models nor the "joint distribution function", which represents statistical knowledge, have to be considered.

An appropriate hierarchy of ***causal*** process models is therefore:

- ***Boolean or structural models*** represent only the existence of parameters and interrelations between them without any information on signs or magnitudes.
  example: a second order proportional transfer function
  $G(s) = b_0 / (a_2 s^2 + a_1 s + 1)$ with $a_i, b_i \neq 0$.
- Additionally, ***sign-based qualitative causal models*** represent the signs of variables and the direction in which each variable affects another one. Any quantitative information is abstracted to the set { -, 0, +}
  example: transfer function $G(s) = [-] / ([+] s^2 + [+] s + 1)$.
- ***Semi-quantitative causal models*** provide, in addition to the information covered by sign-based qualitative models, some rough (absolute or relative) magnitudes of parameters and effects. The semi-quantitative information can be represented by inequality relations (e.g. $a_2 >> a_1 > 0 >> b_0 \geq -100$), value ranges (e.g. $a_1 = [0.8 \ldots 13]$), fuzzy sets or other formalisms.
  example: transfer function $G(s) = [-50 \ldots -100] / ([20 \ldots 150] s^2 + [0.8 \ldots 13] s + 1)$.
- ***Quantitative models*** employ the most detailed numerical and algebraic representations, such as systems of equations and numerical values of parameters.
  example: transfer function $G(s) = -66 / (134.6 s^2 + 7.2 s + 1)$.

This hierarchy is in accordance with Mavrovouniotis and Stephanopoulos [8], but more specific and generally applicable as far as the labels of level 2 and 3, respectively, are concerned.

Boolean models - the most abstract qualitative models - are very useful as 'a priori' information for identification experiments as well as the structural layout of a control system but cannot be used as such for simulation purposes and are therefore not considered in the following discussions. Likewise, the fully defined quantitative models are not the subject of this overview and will not be considered any further. In section 3., the remaining two categories of the above *causal* model hierarchy (***sign-based qualitative causal models*** and ***semi-quantitative causal models***) are complemented by ***heuristics based semi-quantitative models*** to form the three distinct categories in which the surveyed approaches are sorted.

[3] Causal knowledge is knowledge about general relationships between problem solutions and problem features which can be formulated in mathematical equations.

2. Overview of the Different Approaches

In this section, the different modelling and simulation approaches as well as their basic concepts are very briefly introduced. Although this section is particularly helpful to get a general idea of the work that has been done in this field, it could also be skipped initially and used as a reference during the study of section 3.

The publication of Hayes' paper 'The Naive Physics Manifesto' [10] in 1979, in which he proposed constructing a formalisation of a large part of ordinary everyday knowledge of the physical world, was of key importance for the developments in qualitative reasoning. Hayes criticism was that problem solver programs in Artificial Intelligence (AI) addressed only 'toy-problems' and he further detailed the characteristics of the required formalism. Several research projects were triggered off by his ideas which were aiming at the automation of the techniques by which humans reason about the physical world on the basis of very general - or 'qualitative' - information. The following approaches which appear here without a strict order are mostly - either directly or indirectly - inspired by Hayes' concept.

The **'Qualitative Physics based on Confluences'** was developed by De Kleer and Brown [3]. In this theory, qualitative constraints are associated with the components and connections that make up a mechanism. The formalism is based on the concept of confluences, i.e., qualitative differential equations with parameter values abstracted to '-', '0' or '+'. The simulation result is a directed graph of qualitative states that corresponds to the set of all possible sequences of events that can occur from the initial qualitative state ('envisioning', Figure 1).

The **'Qualitative Algebra Q1'** by Williams [11] is an extension of De Kleer and Brown's concept and therefore also based on the availability of mathematical model equations. In Q1, more algebraic operators are allowed than in Qualitative Physics and the abstraction to the qualitative parameters {-, 0, +} is made at a later stage, so that the equations are initially operated on using a symbolic algebra system. Yielding a reduced number of possible future states, the ambiguity of the predictions can be reduced with Q1.

The **'Quantity Lattice'** was designed by Simmons [12] specifically to handle problems with thousands of variables, expressions and inequalities in a computationally efficient manner. The idea was to trade completeness of information for faster, more intuitive deductions. Similarly to the first two approaches, the relationships of parameters need to be specified in mathematical format. In the Quantity Lattice, both simple arithmetic expressions (+,*,-,/) and ordinal relationships ($>,<,=,\neq,\geq,\leq$) are supported.

Forbus' **'Qualitative Process Theory'** [4], which was implemented in 'QPE' [13], has partly evolved from De Kleer and Brown's work and uses the same concept of confluences and

envisioning. Unlike Qualitative Physics, however, Forbus' approach concentrates on modelling physical *processes* (for example flow from A to B) rather than on modelling *components and their interconnections* (like tanks which are connected through pipes). These processes are activated when the appropriate conditions exist and individuals, i.e. physical objects, are present.

Extensions to the Qualitative Process Theory that aimed at enhancing the feature of modularity of this approach have been developed by Falkenhainer and Forbus [14]. To compose a model from different modules, the approach requires a 'domain model', i.e., a library that consists of 'fragments', each describing some fundamental piece of the domain's physics. The user specifies a 'scenario' (important objects, initial conditions and relations) and the simulation environment instantiates the appropriate fragments to yield the scenario model before the simulation is run. A central idea is the explicit statement of the modelling assumptions. The system has therefore been used in conjunction with numerical simulators (e.g. Runge-Kutta) to monitor simultaneously whether any of the modelling assumptions are violated during the simulation. **'SIMGEN'** by Forbus and Falkenhainer [15] requires additionally a mathematical model library that corresponds to the 'domain model'. SIMGEN is used to generate automatically a numerical model from the qualitative as well as semi-quantitative information (scenario). The main idea of SIMGEN is to give explanations and to generate answers to questions about the trajectory of the numerically simulated model. Unlike in NSIM (see below), the automatically generated numerical model is an exact model that does not express the inexactness of the qualitative model in any way.

'Qualitative Process Theory using linguistic Variables' by D'Ambrosio [16, 17] is an extension of Forbus' theory which aimed at overcoming the "severe limitations of QP theory" [17] (i.e., mainly the ambiguity of simulation results) by combining it with the fuzzy theory notion of linguistic variables. Quantitative and semi-quantitative information can therefore be handled to a certain extent in this extension to Qualitative Process Theory.

The **Qualitative Simulation** Approach **'QSIM'** by Kuipers [18, 19, 20, 21, 22] is - like all the above listed formalisms - based on the availability of information about the mathematical relationships between the process variables. Quantitative information is abstracted to {-, 0, +} but ordinal relations with some 'landmark values' are included additionally. Although Kuipers also used the 'envisioning' concept suggested by De Kleer, QSIM features improved facilities to reason about time compared with the other confluences based approaches.

Over the years, QSIM has been extended significantly towards using (semi-) quantitative knowledge: While **'Q2'** by Kuipers and Berleant [23] assigns relatively coarse value ranges to previously labelled landmark values, **'Q3'** by Berleant and Kuipers [24] with its temporal step-size refinement is a significant step towards combining the strengths of both qualitative and

quantitative simulation. In Q3, different degrees of semi-quantitative information can be used, with the output quality being directly related to the quality of the information provided. Unlike Q3, which is concerned with the interpolation of the behaviour at discrete time points, **'NSIM'** (Kay, Kuipers [25]), a third extension to QSIM has specifically been built to increase the precision over the intervals between the time points. NSIM generates for each initial state of the semi-quantitative, qualitative differential equation a set of numerical - normally nonlinear - extremal equations which bound the state variables, and in a second step applies any numerical simulator, such as Runge-Kutta, in order to generate the 'dynamic envelopes', i.e., the trajectories which bound the possible behaviours of the semi-quantitatively defined system. NSIM and Q3 each have advantages and disadvantages with respect to each other and are therefore complimentary approaches that both build on Q2 which in turn builds on QSIM.

'QPC', developed by Farquhar [26], Crawford and Kuipers [27], synthesises the advantages of both the 'Qualitative Process' approach by Forbus [4] and 'QSIM' by Kuipers [21]. It applies and extends the concept of the former expressive compositional model building approach and generates the set of initial conditions suitable for solution by the qualitative simulation engine of the latter approach. The main concept of this compositional modelling approach is that the individual situation is only to be specified in terms of the 'scenario', i.e., objects that are known to be of interest, some initial conditions, and some relations that hold throughout the simulation. Using an appropriately pre-specified 'domain theory' , i.e., a model fragment library that contains for example physical laws (like mass conservation), processes (like liquid flows), devices (like pumps) and objects (e.g. containers), QPC activates automatically the model fragments whose conditions are satisfied either by the initial 'scenario' specification or at any time step during the simulation (therefore 'compositional'). On the basis of the active model fragments, QPC generates constraints that are translated into qualitative differential equations (QDE) as inputs to QSIM and, after computing an initial state, triggers directly the QSIM simulation run.

'SQPC' (Farquhar, Brajnik [28]), the semi-quantitative extension to QPC, handles additionally numeric bounds on magnitudes and monotonic functions, functions specified by look-up tables and dimensional information, which enables very helpful consistency checks. For simulation, it applies QSIM with its semi-quantitative extension Q2; a further extension towards using additionally NSIM is currently being developed.

The **'Order-of-Magnitude Reasoning - O(M) -'** approach by Mavrovouniotis and Stephanopoulos [8] / Mavrovouniotis [9] is different from the others in that it was particularly designed for static modelling for engineering purposes with the facility to include semi-quantitative information. To define the orders of magnitudes that hold amongst the system variables and/or the values of the variables, seven primitive relations such as "much smaller than" and "moderately larger than" are applied. Each of these relations is interpreted with

respect to the location of the quotient of the two compared quantities within an interval. All such intervals - which are disjoint - are defined with respect to a unique parameter chosen according to domain knowledge. The possibility of making efficient use of higher-quality knowledge - possibly even numerical parameter and relationship information - is, however, limited. One of the interesting characteristics of this approach is the possibility to combine a system's mathematical relations with rule-based descriptions. Also, several different relations between two quantities are allowed to coexist.

'Formal Order-of-Magnitude Reasoning, FOG' developed by Raiman [29] is similar to Mavrovouniotis' O(M) approach in its method of handling semi-quantitative information. Having been suggested before the above approach, it does, however, not have the latter extended features of O(M) mentioned above.

The **Fuzzy Qualitative Simulation Approach 'FuSim'** by Shen and Leitch [30, 31] adopts the approach taken by QSIM, but, for the purpose of semi-quantitative reasoning, applies fuzzy numbers rather than symbolic landmarks. Using fuzzy sets as representations for both quantities and strength annotations to mathematical relationships between variables allows for the generation of temporal durations.

'Mycroft' (Coghill, Chantler [32]) is a qualitative reasoning framework that allows for different ways of performing a qualitative reasoning task within the same environment. It combines the best features of FuSim with those of other approaches, which are not explicitly considered here, in a single approach (for more details see Coghill, Chantler [32]). The above characteristic of FuSim applies therefore likewise to Mycroft. However, Mycroft requires causality information in addition to the equation set and can either be run semi-constructively or constructively[4] which makes it unique among the compared approaches.

Completely different from the above qualitative reasoning approaches that are rooted in Hayes' Naive Physics concept is the application of *heuristic* **If - Then Rules** (overview by Puppe [6]). Mathematical relations are not necessary for this approach - although they could be integrated. The main idea, however, is to collect unstructured knowledge from experience for model building purposes. The general system knowledge is collected in a rule base while facts about any specific situation form the fact base. Rules are evaluated or 'fired' if the facts match their condition - or 'IF' - part. Only one rule is fired at a time and different algorithms exist that determine which rule is fired in case that the condition part of several rules is fulfilled. The result of a rule evaluation is added to the fact base and a new rule evaluation cycle commences by comparing the updated fact base with the condition parts of the rules. The rules are recursively 'chained' in this manner until no more rule conditions are fulfilled. For multiple-

[4] The term 'constructive' refers to whether the algorithm uses the system constraints to generate unique output states (i.e., constructive), or to filter out the inapplicable among previously generated states (i.e., non-constructive).

phase simulation of such models, time-dependent rules are triggered additionally by a global clock.

Fuzzy Modelling, based on Zadeh's notion [33], covers to a certain extent the functionality of purely rule-based modelling, although the rules are normally not chained. The main characteristic, however, is the facility to handle uncertainty of parameters and decisions in addition. Fuzzy set theory, the basis of fuzzy modelling, is a generalisation of conventional ('crisp') set theory in that it allows for graded set-membership in addition to full and non-membership and by this means deals with uncertainty. Also, the fuzzy approach is aimed at dealing with complex multivariable systems on the basis of relatively limited information.

Likewise independent of Hayes' concept, the **classical system theory** applies qualitative models in the case of insufficient system knowledge. Different general behaviour and stability analyses of systems are possible on the basis of at least structurally defined mathematical relationships. In [1], Lunze introduces simple abstraction operators which are applied to a quantitative state space model to show the close relationship between the abstracted model representations in qualitative reasoning and conventional system theoretical models. Similarly to most of the above approaches, qualitative simulations of continuous processes are carried out using discrete approaches - for example **Automata** [34, 2] or **Petri-nets** (more specifically state machines[5]) [35, 36]. For simulation purposes, however, a mathematical description in the form of abstracted differential equations is normally not necessary. To specify a process model, both approaches require the set of possible system states, possible input settings, an initial condition and an input sequence. Additionally, a transition function must be specified for automata while for Petri-nets, a net topology as well as input conditions for 'labelling' the transitions between the nodes or 'places'[6] must be defined. Both approaches can generate all possible qualitative trajectories, but mixed with spurious solutions. Automata feature a coarse reference to the time scale while the Petri-net approach requires special extensions to represent information about temporal relations. Due to their similarity, state machines and automata can directly be translated into each other.

An interesting suggestion to exploit different kinds of knowledge was made by Kluwe, Krebs, Lunze and Richter [37, 38, 39] who introduced a **Three-Layer Process Model** in which quantitative information is represented in the form of differential equations, while qualitative knowledge is exploited both by a Petri-net and heuristic rules. The ambiguities of the central component, the Petri-net, are resolved in this approach either with the help of the heuristics or by the sets of differential equations that are individually assigned to every place (i.e., qualitative state) in the Petri-net. This approach is particularly useful for large scale systems

[5] State machines are a specific class of Petri-nets, in which every transition has exactly one predecessor and one successor.
[6] Each place in the Petri-net represents a semi-quantitative state in terms of ranges.

where sufficient quantitative mathematical information about the process in its various conditions can be acquired.

3. Qualitative Models - the Comparison of their Characteristics

In this section, the main characteristics of the above introduced qualitative modelling approaches with respect to the application to modelling and simulation in control engineering are discussed. According to their level and kind of abstraction, the approaches are firstly sorted into the three distinct categories, which have been introduced in section 1: ***sign-based qualitative causal models***, ***semi-quantitative causal models*** and ***heuristics based semi-quantitative models***.[7]

Sign-based causal **approaches on the basis of mathematical relations:**

- 'Qualitative Physics based on Confluences' (De Kleer, Brown [3])
- 'Qualitative Process Theory', implemented in 'QPE' (Forbus [4])
- the 'Qualitative Simulation Approach QSIM' (Kuipers [18, 19, 20, 21, 22])
- the 'Qualitative Physics Compiler QPC' (Farquhar [26, 27])

Semi-quantitative causal **approaches:**

- the 'Qualitative Algebra Q1' (Williams [11])
- the 'Quantity Lattice' (Simmons [12])
- 'Qualitative Process Theory using linguistic Variables' (D'Ambrosio [16, 17])
- 'SIMGEN', an extension to Qualitative Process Theory (Forbus and Falkenhainer [15])
- 'Q2', 'Q3' and 'NSIM' the semi-quantitative extensions of QSIM, (Kuipers and Berleant [23], Berleant/Kuipers [24], Kay/Kuipers [25], respectively)
- the 'Semi-Quantitative Physics Compiler, SQPC' (Farquhar and Brajnik [28])
- 'Order-of-Magnitude Reasoning O(M)' (Mavrovouniotis, Stephanopoulos [8] / Mavrovouniotis [9])
- 'Formal Order-of-Magnitude Reasoning, FOG' (Raiman [29])
- the 'Fuzzy Qualitative Simulation Approach FuSim' (Shen, Leitch [30, 31])
- the 'Qualitative Reasoning Framework Mycroft' (Coghill, Chantler [32])
- 'Petri-nets' and 'Automata' (Lunze [36], Bredebusch/Lunze/Richter [35], Lunze [2], Hopcroft/Ullman [34])
- combined quantitative-qualitative modelling using a 'Three-Layer-Model' (Kluwe, Krebs, Lunze, Richter [37, 38, 39])

[7] This is just a simplified classification for the purpose of the following evaluation. The different approaches are classified according to their main concept.

***Heuristics* based semi-quantitative approaches:**

- If - Then Rules (see overview by Puppe [6])
- Fuzzy Modelling (based on Zadeh's notion [33])

Although the above *heuristics* based approaches are normally not considered as qualitative modelling approaches in the narrow sense, they are included here because they fall into the group of semi-quantitative modelling approaches. In [40], Sugeno and Yasukawa argue similarly and state that "there are small distinctions and big similarities between fuzzy modelling and qualitative reasoning". They stress the advantages of fuzzy modelling in practical applications and suggest a fuzzy based approach to qualitative modelling on the basis of numerical process data.

The difference between the *semi-quantitative causal* approaches which employ the fuzzy theory and the *heuristics*-based fuzzy modelling approach is that the former make only use of 'linguistic variables' as a means of representing uncertainty in the parameters and relationships while the latter additionally applies fuzzy If-Then rules to represent the process models altogether. Rather than applying rules, the *semi-quantitative causal* approaches are still based on mathematical system equations like all other *causal* approaches.

In the following discussion, the members of each of the groups (*sign-based causal, semi-quantitative causal* or *heuristics* based) will often be addressed collectively. For more information on the approaches whose characteristics are not explicitly discussed in either of the below sections, please refer to **Table 1**, which gives a complete overview with ratings.

This study is structured in aspects **a)** to **g)**, according to the most important characteristics for modelling approaches with respect to the application in control engineering:

a) Ambiguity, Accuracy and Precision of Predictions
b) The possible Complexity of the modelled Process
c) Temporal Aspects
d) Using the available Knowledge
e) The Model Formulation
f) Combining qualitative and quantitative Models in a Simulation
g) Utilisation of the Methodologies

a) Ambiguity, Accuracy and Precision of Predictions

While 'precision' refers in the context of qualitative and semi-quantitative simulation to the tightness of bounds around a predicted behaviour, accuracy is concerned with the generation of all possible behaviours of the partially defined models. Ambiguity normally occurs whenever possible behaviours are 'hidden' between spurious ones.

The *sign-based causal* approaches, which work on the abstraction level of the set {-, 0, +} (i.e., confluences based approaches), allow only for very coarse predictions of possible future states (Figure 1). Inevitably, the absence or neglection of any numerical information results in ambiguous predictions of the model's behaviour. This is acceptable in some applications of qualitative reasoning, but it is inappropriate for simulation in the control engineering domain as it is for most other technical applications.

The *semi-quantitative* extensions like Q1 (Williams [11]) and Q2 (Kuipers and Berleant [23]), which integrate some quantitative information in the form of parameter values and measured data, can reduce, albeit not remove, the ambiguity. Their precision is, although improved from results like "somewhere between 0 and +∞" to coarse value ranges, still very low. With D'Ambrosio's linguistic extension to the Qualitative Process Theory [16], both the number of spurious behaviours is reduced and the precision of the predictions is improved. Although D'Ambrosio's approach appears to be promising, many of the original ambiguities of QP Theory remain, as he detailed in the conclusions of his work.

Compared with these early extensions to confluences based techniques, the model-based approaches which are purpose-built for (semi-) quantitative knowledge - particularly the O(M) technique [8], FuSim [30] and Mycroft [32] - are more sound in the different aspects[8] of applying this information. Similarly to D'Ambrosio's approach, FuSim and Mycroft apply a fuzzy quantity space which is particularly suited to the problem domain since the semi-quantitative knowledge can not only be specified as value ranges but also uncertainty can be expressed through the graded membership of fuzzy sets. At each time increment, the simulation input and output of these two approaches is defined in terms of fuzzy sets. Using fuzzification and defuzzification, the latter approaches could be extended to handle and to yield numerical input and output data, respectively. Although such an extension that trades accuracy for precision would in a way contradict the genuine idea of these approaches and also imply an unrealistic exactness, it could enable a more flexible use of the models which will be discussed under aspect **f)**.

[8] For example the propagation of (semi-) quantitative information, the precision of predictions using more aggressive, heuristics based conclusions and the renunciation of abstracting the provided numerical knowledge.

An example for an approach that sacrifices accuracy for precision is SIMGEN, which generates numerical models from qualitative descriptions using a mathematical model library, a semi-quantitative domain library and a matching algorithm.

With respect to ambiguity, accuracy and precision, Q3 and NSIM, the further extensions to QSIM and Q2, have arrived at a level that is so far unparalleled by any other causal approach: while still retaining the accuracy of the underlying QSIM, the precision of the predictions is here directly related to the precision of the initial conditions[9] . Q3 has also been proven to converge, i.e., with decreasing simulation step sizes, the quality of fit with respect to the actual process behaviour increases under the assumption of precise initial conditions.

The Petri-net and automata approaches have been shown to provide sufficient information to allow for the design of qualitative controllers [2, 35] which are assumed to have only coarse measurement information available. Nevertheless, these process models yield only very general information when applied to simulation, which is due to their remaining ambiguity. In [35], it was shown that using state machines, the proportion between spurious and real solutions cannot be improved by increasing the number of qualitative states. The simulation results are comparable to some of the earlier semi-quantitative approaches in Artificial Intelligence (e.g. Q1, Q2). The combination of Petri-nets with quantitative equations as well as heuristics in the Three-Layer Model is more appropriate for simulation purposes although the sudden switching of system characteristics due to fired transitions in the Petri-net could lead in parts to unrealistic trajectories.

Normally, *heuristics* based models predict unambiguously only a single behaviour in each situation; their accuracy in the above sense is therefore often low while their precision varies significantly, depending on the type of implementation. However, a fine discretisation of explicitly considered states yielding more precise predictions can lead to an exploding rule-base. This disadvantage applies particularly to purely rule-based models and can largely be overcome with the fuzzy approach to *heuristic* modelling. The quality of fuzzy models depends additionally on the definition of the fuzzy membership functions. Within this comparison of qualitative and semi-quantitative modelling approaches, however, the simulation of fuzzy models stands out with respect to precision. The defuzzified, numerical output values (Figure 2) can be close approximations of the real process behaviour. These predictions are therefore in a sense false, but close to reality (i.e., precise), whereas for example the confluences-based qualitative models predict the real behaviour (accuracy), but only in very broad terms and hidden between spurious behaviours (ambiguity). Although QSIM with all its extensions (Q2, Q3, NSIM) is advantageous to the fuzzy approach with respect to accuracy, it can only catch up with respect to ambiguity and precision when the conditions are numerically fully defined.

[9] In Artificial Intelligence, this characteristic is known as 'stability'.

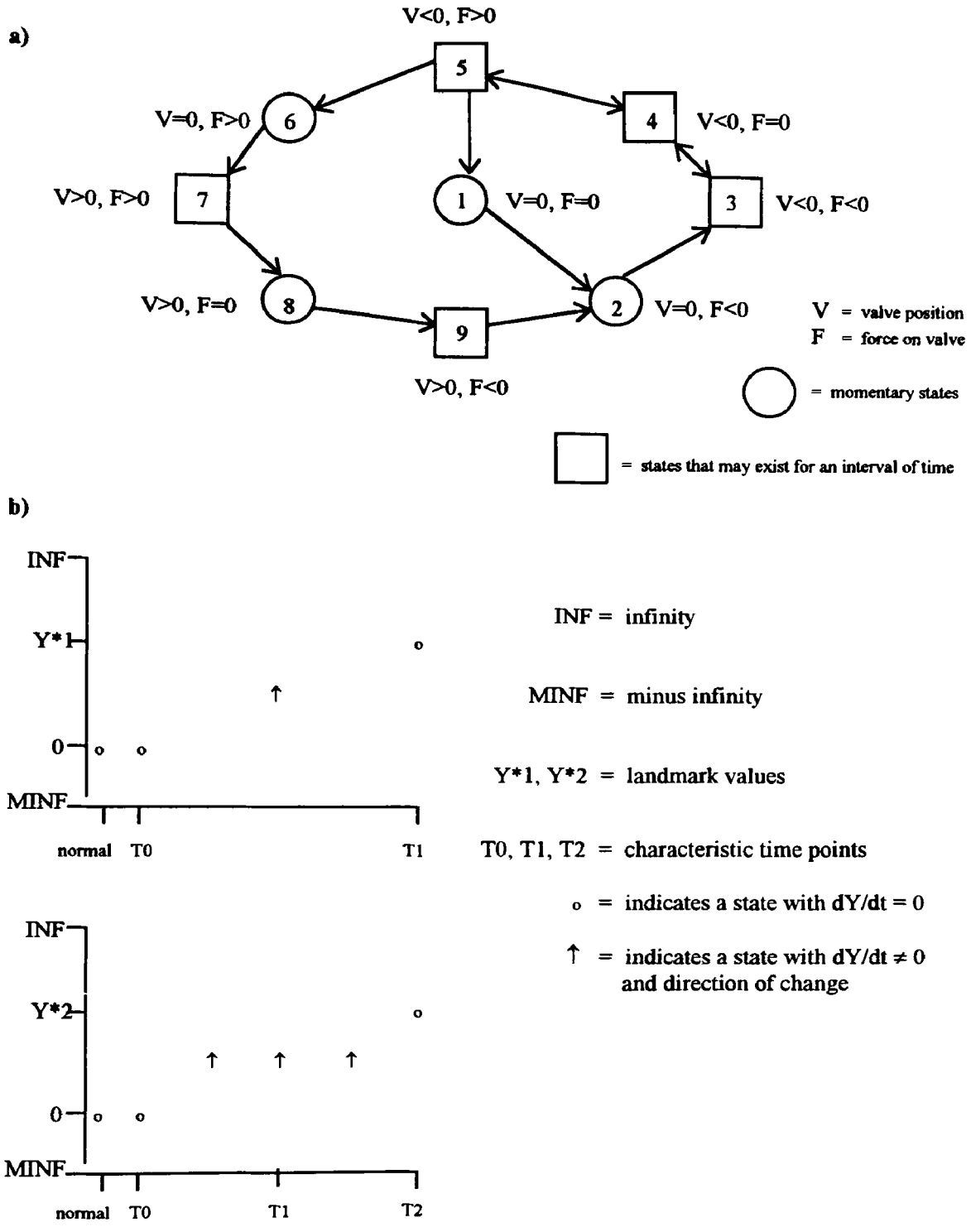


Figure 1: Envisioning - a) State Transition Diagram from an example in [3], (©1984 Elsevier Science B.V.[10])
for a pressure-regulator with continuing input signal, mass, spring and no friction
b)Time-Plot of a Step Response in QSIM [20], (©1988 Elsevier Science Ltd[11])
possible predictions for a system of two first-order processes in series

[10] Reprinted from Artificial Intelligence, vol. 24, De Kleer and Brown: "A Qualitative Physics based on Confluences", pp. 7-83, ©1984, with kind permission from Elsevier Science B.V., P.O. Box 521, 1000 AM Amsterdam, The Netherlands
[11] Reprinted from Comput. Chem. Engng., vol. 12, Dalle Molle, Kuipers, Edgar: "Qualitative Modeling and Simulation of Dynamic Systems", pp. 853-866, ©1988, with kind permission from Elsevier Science Ltd, The Boulevard, Langford Lane, Kidlington OX5 1GB, UK

b) The possible Complexity of the modelled Process

Modelling is closely related to abstraction, since every model is an abstraction of the real system. The mutual implications between system complexity and abstraction levels is briefly summarised before the qualities of the different modelling approaches are discussed. In the context of qualitative and semi-quantitative modelling, two types of abstraction and their implications on the modelling of complex systems must be distinguished:

(I) - The more generally applied meaning of 'abstraction' refers to taking a step back and considering a complex system from a more global point of view, neglecting the details of the individual components that make up the system. Modelling approaches that allow for this type of abstraction can deal quite easily with complex systems because the model complexity is not directly related to the system complexity. While approaches that build on physical balance equations are unable to abstract in this sense, *heuristics* based approaches are particularly suitable for this type of abstraction.

(II) - The other understanding of 'abstraction' which is mostly used in qualitative modelling refers to coarse signals and process models whose parameters in the equations are not numerically exactly known but only in terms of value ranges, orders of magnitudes, or, in the most 'abstract' case, in terms of their signs.

For increasing system complexity, this second type of abstraction imposes a major limitation - particularly on some of the *causal* approaches:

It can generally be stated that the more abstract (II) the modelling level, i.e., the less numerical information is available or used, the more possible qualitative trajectories can be distinguished for a single set of initial conditions. Also, it is inevitable that with an increasingly abstract (II) simulation, the amount of additional, spurious behaviours generated by the simulation environment increases. All these problems increase dramatically with the complexity of the modelled process unless the modelling approach allows for abstraction in the first sense, taking a more global view of the system. Although all *causal* approaches have to fight with the same problems, they differ in the attempts and success in keeping the complexity dependent ambiguity manageable.

Bearing this in mind, it is therefore not surprising that the most abstract approaches that cannot handle numerical information are very limited with respect to possible complexity. Although these confluences-based *causal* approaches (a most notable example being QSIM, which was repeatedly applied to comparably complex situations) cope respectably, they are inappropriate for modelling in control engineering since they cannot make use of any numerical or semi-quantitative information that is normally available in this domain.

Even many of the *semi-quantitative causal* approaches, like QSIM together with the post-processor 'Q2' have their limitations when the number of remaining spurious predictions becomes unmanageable (Dalle Molle [41]).

Among the *semi-quantitative causal* approaches that handle the information more efficiently and largely avoid increasing ambiguity with increasing complexity are the order-of-magnitude approaches, FuSim and Mycroft, as well as QSIM with Q3 and NSIM and the system science approaches automata and Petri-nets. The most suitable approach for complex situations, however, appears to be the Three-Layer-Model provided that sufficient quantitative mathematical information about the process in its various conditions can be acquired.

The rather low rating of SIMGEN in Table 1 is not related to the generation of spurious behaviours since it performs essentially a numeric simulation, but it is due to the fact that it relies entirely on a mathematical component library which can easily reach its limits - particularly the more complex systems become.

Using *heuristics* based approaches, the modeller is free to decide on the type (I) abstraction level of the process model, as it was mentioned above. Even very complex processes can therefore be modelled in a simple way without increase of ambiguity if only general predictions for the main parameters are needed. Also, there is no **direct** correlation between process complexity and possibly extensive modelling effort, but mainly between the required precision of the simulation and the modelling effort. In the case, however, that a significant number of discrete states needs to be considered in purely rule-based process modelling in order to yield satisfactory precision, an 'exploding' rule base (see aspect a)) could result. The fuzzy based approach with its nonlinear interpolation facility for continuous output states does not suffer so much from such problems as it requires only a comparably coarse 'grid' of system states to be considered in the rules.

c) Temporal Aspects

The *causal* qualitative multiple-phase simulation techniques are mostly based on a similar notion of time: time is composed of intervals that may be related in different ways, like one interval being before, after or equal to another. Likewise, 'histories' are normally applied to represent how things change through time. A particularly important characteristic is the projection of 'trends' for the purpose of coarse temporal predictions.

Despite this common ground among the model-based (*causal*) approaches, there are several differences. Particularly the early works, for example by Forbus, De Kleer and Brown, are limited in their ability to reason about time, as Williams [42] points out. He developed therefore his *semi-quantitative* 'temporal constraint propagator' (TCP), which is based on the notion of 'value history' and was implemented in Simmon's 'Quantity Lattice'. The TCP improves the

maintenance of a consistent partial order of time points by assigning durations to the qualitative states and answers queries about relationships between time points. The consideration of time delays and temporal reasoning for feedback systems are among Williams' particular contributions.

One of the more advanced approaches among the *sign-based causal* methodologies is 'QSIM' in that it uses a 'standard' mathematical model of time [21]. Also, QSIM differs from the other approaches in allowing new landmarks to be discovered during the qualitative simulation, thus creating additional time points[12] as new qualitative distinctions on the time scale. This yields better information on characteristic states such as increasing, decreasing or stable oscillation. Nevertheless, the lack of a representation for time delays is a deficit of QSIM. Since QPC actually employs QSIM, it shares most of these characteristics.

Despite their advances, neither TCP nor QSIM allow for truly dynamic considerations because of their inability to determine the time needed for state transitions.

Among the *semi-quantitative causal* approaches, neither the order-of-magnitude approaches take temporal aspects into account since they perform only algebraic reasoning, nor does D'Ambrosio's fuzzy logic extension to qualitative process theory provide any information on temporal durations. In fact, only few of the approaches feature good representations of dynamic behaviour: Both FuSim (Figure 2) and Mycroft take account of the time that an output of interest remains in a distinguished state as well as of the time for its transition to the next distinguished state (i.e., persistence- and arrival time, respectively). Also, the iterative refinement of time steps in Q3 as well as the three approaches NSIM (Figure 2 c), SIMGEN and the Three-Layer-Model which apply in parts numerical simulators yield good results.

With respect to the simulation of *heuristics* based models, temporal aspects are differently included in purely rule-based and fuzzy approaches. Rule-based multiple-phase simulation applies a global clock that advances the simulation process in discrete, numerical intervals of time and initiates state changes by the activation of time-dependent rules. The length of the intervals is application dependent and may be constant or variable. For the resulting new parameters of the system, the time-independent rules are evaluated afterwards and further parameter values are calculated. The clock is advanced and this sequence is iterated as long as time-dependent rules are triggered. With this concept of rule-based multiple-phase simulation, a determination of time required for state changes is therefore possible.

[12] Distinguished time points are those points where something important happens to the value of the function, such as passing a landmark value or reaching an extremum [21].

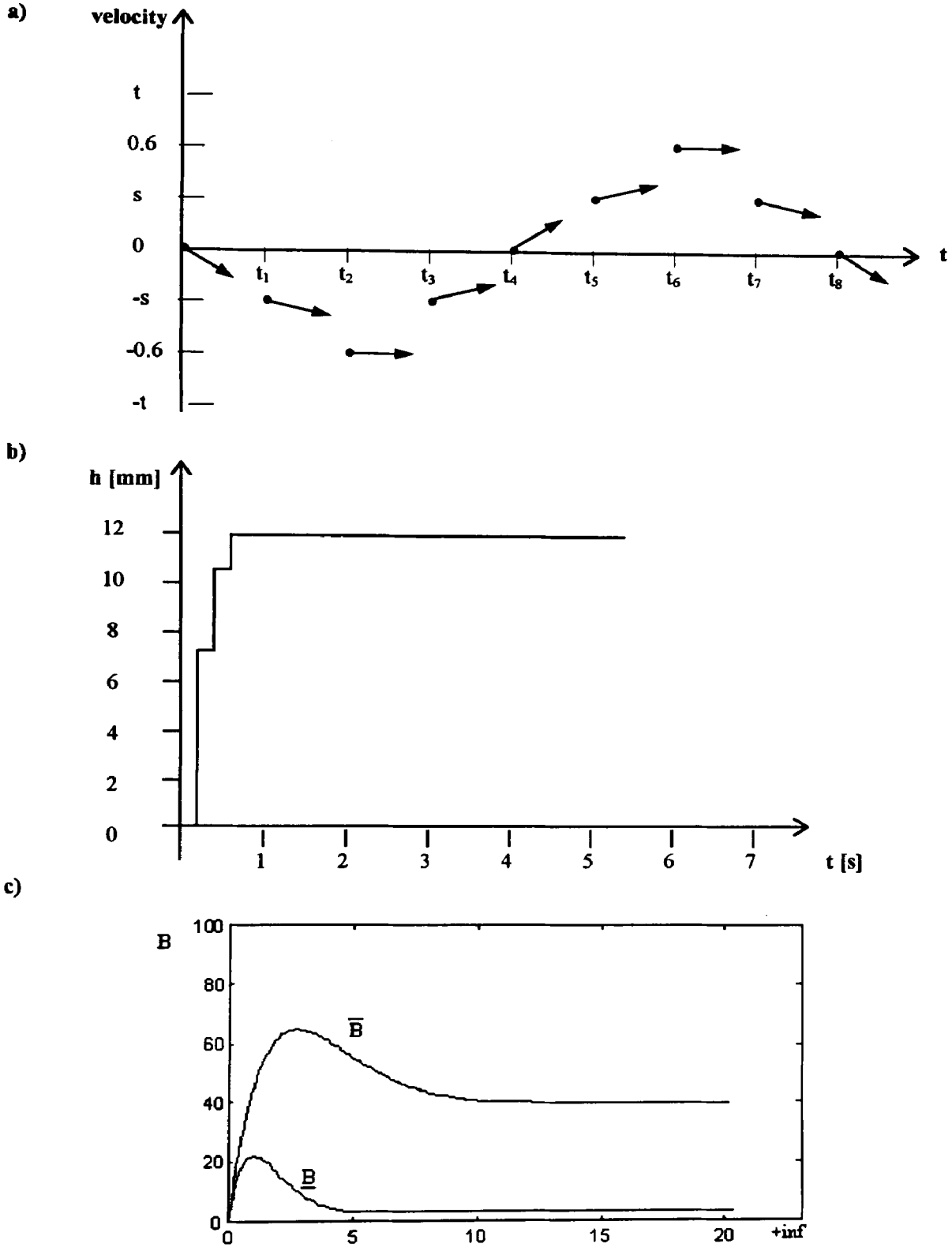


Figure 2: Results of semi-quantitative simulation -

a) Time plot in FuSim [30], ©1993 IEEE[13]: velocity of a 'mass on a spring' system

b) The step response of a fuzzy model: the output derivative is fed back as additional input

c) NSIM output [25], ©1993 AAAI[14]:

Dynamic envelopes defining the lower bound $\underline{B}(t)$ and the upper bound $\overline{B}(t)$ on $B(t)$

---

[13] Reprinted from IEEE Trans. on Syst. Man and Cybernetics, vol. 23, Shen and Leitch: "Fuzzy Qualitative Simulation", pp. 1038-1061, ©1993, with kind permission from IEEE, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, U.S.A.

[14] Reprinted from the Proceedings of the Eleventh Nat. Conf. on AI (AAAI-93), Kay and Kuipers: "Numerical Behavior Envelopes for Qualitative Models", the permission has been applied for at AAAI, 445 Burgess Drive, Menlo Park, CA 94025-3496, U.S.A.

Although this has not yet been tried, the above described discrete multiple-phase simulation approach could likewise be applied to fuzzy models to handle additionally uncertainty with respect to parameter values. The most common technique to yield a dynamic fuzzy model is to apply the derivatives or integrals of input or output variables as additional inputs to the fuzzy system. Using first order derivatives as additional inputs, however, results normally only in a very coarse, 'dynamic-like' behaviour (Figure 2), while higher order derivatives become unmanageable for 'manual' modelling (i.e., formulating the rule base and defining membership functions). Only process data based identification approaches can use the potential of higher order derivatives. Nevertheless, other approaches have been applied to consider dynamic aspects in the normally static fuzzy concept. Still though, these concepts are either not truly dynamic approaches[15] or not generally applicable[16].

d) Using the available Knowledge

De Kleer and Brown stated that by taking the qualitative approach, an often significant amount of knowledge loss cannot be avoided [3]. In contrast to this understanding, the following statement by Mavrovouniotis [8] describes best the idea of applying any of these modelling techniques in control engineering:

"One of the motives for using AI-methods is the desire to apply as much of the available knowledge as possible, despite the disparity in the forms of the knowledge involved."

The keywords in this sentence are "as much as possible" and "available".

In this comparison of modelling approaches, the aspect "available knowledge" also includes the question for the minimum of required knowledge for each of the approaches. In the domain of process modelling for control engineering in industry it is assumed here that at least some sort of semi-quantitative information is readily available, while the mathematical system equations are not always at hand.

By definition, *heuristics* based models enable the usage of experience gained with the real process. Also, they allow for the integration of other different types of knowledge - even mathematical relations and, most notably, quantitative and semi-quantitative knowledge which is usually available in the engineering domain. Although *heuristics* based models are not very efficient in exploiting *causal* information, they fulfil the aspect of using the available knowledge overall very well.

[15] For example the programming of standard trajectory patterns [43, 44].
[16] For example averaging locally valid trajectories of linear dynamics to yield a global system behaviour (Sugeno, Kang [45]) is often erroneous as soon as oscillations occur because of cancellation effects due to phase differences.

The *sign-based causal* approaches which neglect any (semi-) quantitative knowledge clearly do not comply with Mavrovouniotis' above cited maxim.

In general, the *causal* approaches are only applicable if the main physically meaningful relations are known - additionally, some of the approaches (like QSIM and Mycroft) require explicit causality information. It is this characteristic that implies the major disadvantages of 'deep' knowledge[17] representations. With respect to the *causal* modelling approaches, it is interesting to note that hardly any attempts were made to overcome these disadvantages, because the acquisition of the 'deep' knowledge was explicitly seen as being outside their scope.

Among the *semi-quantitative causal* approaches, Mavrovouniotis' O(M) approach is a step towards more flexibility as it allows for integrating some *heuristic* relationships in addition to mathematical ones. This advantage, however, comes at the expense of loosing the strict causality. Another approach that combines different forms of knowledge is the Three-Layer-Model, although its disadvantage is the requirement of a significant amount of quantitative mathematical information.

e) The Model Formulation

Heuristics based knowledge is normally available and intuitively understandable as the rules are usually formulated in plain English. However, even the application of the most recent graphically oriented fuzzy shells requires the theoretical understanding of the modelling methodology. In fact, fuzzy modelling can be highly complex with a wealth of parameters to set and algorithms to choose from. Without guidance, the inexperienced modeller will find it very difficult to translate his/her unstructured, practical experience into a structured rule base. The main problem with respect to purely rule-based modelling is closely related to what was discussed under aspects **a)** and **b)**: the need for often highly complex rule-bases makes the model formulation very cumbersome and error-prone.

The *model-based (causal)* approaches often necessitate a very thorough theoretical understanding of their underlying concept and a specific descriptive language or input format must often be learned. To maintain causality, the formulation of many 'trivial' constraints is in some cases obligatory which makes the formulation complex and cumbersome [7]. Dalle Molle concluded in his PhD-thesis [41] on the investigation of the modelling and simulation of chemical processes using QSIM, that qualitative model building was not a well-defined procedure but something of an art. These problems are not uniquely related to QSIM but apply likewise to the other *causal* approaches as Rajagopalan [46] indicated. More recent research at the University of Texas (e.g. Farquhar, [26]) aimed therefore at the simplification and partial

[17] 'Deep' knowledge refers to causal knowledge, as opposed to 'shallow' (i.e., heuristic) knowledge which does normally not represent the causal nature of an observation.

automation of the model building procedure but resulted only in quite limited improvements. Farquhar's simplifications are based on the application of fragment libraries for different domains. Experience from botany and chemical engineering which was gained by other researchers applying QPC showed, however, that "libraries of realistic model fragments ... are extremely difficult to construct" [26]. The library based approach, which is not related uniquely to any particular modelling approach[18], is certainly one of the most effective means of simplifying modelling. Nevertheless, it is important to consider that any library is limited and simplified extensions to such libraries should be possible. Since hardly any libraries for industrial process modelling are developed so far, the rating in Table 1 considers both the use of a library (if applicable) and the ease of its extension.

In Mycroft, a macro function is applied which translates the textual model specification into the correct format. Although this greatly simplifies the model definition it still requires some familiarity with the required textual format.

f) Combining qualitative and quantitative Models in a Simulation

Ideally, it should be possible to compose a structured process model arbitrarily from components of different model types. The direct implication of combining a qualitative process model for example with a conventional transfer function for simulation purposes is the need for precise numerical inputs and outputs of the individual components.

According to the discussion in **a)**, fuzzy models with continuous numerical input and output variable ranges fulfil this requirement as well as the Three-Layer-Model and SIMGEN, while Q3 and NSIM yield normally less precise results in favour of accuracy. If used within a general purpose simulation environment, however, both the fuzzy and the Three-Layer-Model approach are easier to interface with conventional models than SIMGEN or Q3/NSIM.

It would also be possible to extend the order-of-magnitude approaches and particularly the fuzzy quantity space based approaches FuSim and Mycroft so that numerical outputs are obtained, while the use of numerical input values is anyway unproblematic. Although the linguistically extended Qualitative Process Theory [16] is also based on a fuzzy quantity space, the defuzzification of its outputs in order to obtain numerical values would not be sensible due to the ambiguities of this approach.

It must be stressed here that precise numerical output values of any qualitative approach always bare the risk of being misinterpreted as exactness, although they are actually approximations whose quality depends on the amount of knowledge provided as well as the algorithm used.

[18] Among the qualitative modelling approaches, basic libraries have been developed for example by De Kleer/Brown and Forbus.

The only way to combine the remaining *causal* qualitative models with other model types would be in series, with the *causal* qualitative model always at the end.

g) Utilisation of the Methodologies

This aspect is primarily concerned with the availability of program modules, code or implementation recipes for the different qualitative modelling and simulation methodologies. As a secondary factor, the ease of implementation is also considered.

Heuristics based approaches are featured in many commercially available programs like expert systems or fuzzy shells. Nevertheless, only few purpose built frame-based expert systems and fuzzy process modelling tools for control domain relevant simulation of industrial processes exist[19]. Among the more recent programs which allow for the specification of fuzzy models as well as their simulation in arbitrary combinations with conventional process models, Matrix$_x$™ and MATLAB™ (with fuzzy toolbox) should be mentioned. Another advantage of the fuzzy modelling approach is its detailed documentation in the literature as well as its relative simplicity which allows for quite quick and simple individual implementations of the whole modelling and simulation approach, if necessary.

Among the system science based *semi-quantitative causal* approaches, only Petri-net modelling and simulation programs are commercially available. Individual implementations of Petri-nets - possibly within general purpose simulation environments - are laborious but possible due to the widespread expertise and documentation. Although also based on well established approaches, the Three-Layer-Model is more difficult to implement individually as the interplay between the components is of key importance. It is therefore more sensible to build on the extensive work on the implementation of "DEMSI", a simulator for the Three-Layer-Model, which has mainly been carried out at the Technical University of Hamburg-Harburg[20]. The development of "DEMSI" has particularly focused on its online application in advisory systems for complex plants.

The situation is different for the artificial intelligence-rooted *causal* approaches as these are not based on a standard concept. In order to make use of them it is necessary to fall back on the existing programs since they consist of comprehensive sets of reasoning algorithms which resulted from individual research projects. Since most of the existing programs are normally only research implementations which were used for validation purposes, they have often never been in a very usable form or the code has been abandoned after the research projects have

[19] For an overview see Linkens [47] and Puppe [6].
[20] For further information on DEMSI please contact Prof. Dr. Jan Lunze or Dr. Henrik Richter, Technical University Hamburg-Harburg.

been finished. This aspect reduces dramatically the group of potentially applicable approaches to QPC, SQPC, Mycroft and QSIM, including its extensions Q2, Q3 and NSIM.

The QSIM program has been maintained and updated and is available in LISP source code via 'ftp' from the file server at the University of Texas at Austin[21], together with its extensions. A good deal of experience in compiling and installing such systems is, however, required. The same applies to QPC and SQPC[22].

Mycroft is currently being finalised in Common LISP code and will be available via 'ftp' from Heriot Watt University in Edinburgh[23]. Similarly to QSIM, the installation will probably require some experience. A re-implementation of Mycroft in C++ which is also being considered at Heriot Watt University could simplify the installation and improve both the portability and speed and therefore further increase the attractiveness of this interesting approach for a broader community.

4. Summary and Selection

For a quick reference, the results of the discussion in the preceding section are summarised in the following Table 1. The ratings which range from 'very poor' (- -) to 'very good' (++) should give a general feel for the comparison between the approaches with respect to the considered characteristics a) - g). From this table, it is quite obvious that *sign-based causal* approaches are unsuitable for practical control engineering purposes which is generally due to their over-abstraction of readily available (semi-) quantitative information.

Despite its deficiencies with respect to dynamics, the fuzzy based modelling approach turned out to be advantageous in almost all aspects. However, since the representation of process dynamics is generally not a strength of the qualitative and semi-quantitative modelling approaches, fuzzy modelling is not really an exception as it was discussed above. Overall, this approach is therefore the first choice.

Among the *semi-quantitative causal* approaches, the Three-Layer-Model, Mycroft and QSIM with Q3 and NSIM are the most appropriate approaches for application in control engineering which have not only advantages and disadvantages with respect to each other, but in particular with respect to the fuzzy approach. Depending on the individual emphases, any of these three *causal* approaches could be a sensible complement to the fuzzy modelling approach.

---

[21] Please contact Prof. Dr. Benjamin Kuipers, University of Texas at Austin, for information on the terms of usage.
[22] Please contact Prof. Dr. Adam Farquhar, Stanford University.
[23] Please contact Dr. George M. Coghill, Heriot Watt University / Edinburgh, for further information.

	semi-quantitative causal												
	QSIM incl. Q2	QSIM, Q2,Q3, NSIM	SQPC	Qlattice	Q1	SIMGEN	O(M)	FOG	QProc + fuzzy	FuSIm	Mycroft	PetriNet auto-mata	Three-Level Model
a) precision, accuracy ambiguity	o	++	o	o	o	+	+	+	o	+	+	o	+
b) possible complexity	o	+	o	+	o	o	+	+	o	+	+	+	++
c) temporal aspects	o	+	o	o	o	+	--	--	-	+	+	o	+
d) using available knowledge	-	-	-	-	-	-	+	-	-	-	-	-	o
e) model formulation	-	-	-	-	-	-	-	-	-	o	o	o	-
f) combination with quantitative models	--	o	--	--	--	+	-	-	--	-	-	--	++
g) utilisation	o	o	o	--	--	--	--	--	--	--	o	+	o

Table 1 a)

	sign-based causal			heuristics-based	
	QPhys	Qproc (QPE)	QPC	rule based	Fuzzy
a) precision, accuracy ambiguity	-	-	-	+	++
b) possible complexity	-	-	-	o	+
c) temporal aspects	-	-	o	o	o
d) using available knowledge	--	--	-	+	++
e) model formulation	-	-	-	-	o
f) combination with quantitative models	--	--	--	o	++
g) utilisation	--	--	o	+	++

Table 1 b)

5. Conclusion of the overview

In this survey, the ideas behind qualitative modelling which is being pursued by several research communities have been clarified. Different abstraction levels of qualitative models as well as the most commonly used knowledge types - *causal* and *heuristic* knowledge - have been illustrated in the introductory section in order to specify the three categories into which the considered qualitative modelling approaches have subsequently been split: *sign-based causal* approaches, *semi-quantitative causal* approaches and *heuristics based semi-quantitative* approaches. These three groups of approaches cover the whole range of existing qualitative modelling approaches.

Although some of the main techniques within each class have been introduced and compared, this overview does not claim to be complete with respect to individual approaches. Also, it should again be stressed that all comparisons and ratings are made with the application to modelling and simulation in control engineering in mind. Therefore, it is well possible that approaches which collected mainly bad ratings (Table 1) have characteristics that are exceptionally well suited to other tasks. SIMGEN, for example, has been particularly designed for self-explanatory simulations: any question about the system behaviour of the type "What if ...?" or "Why ...?" can be asked and is automatically answered by the program, which makes it an interesting approach for tutorial purposes.

Some of the earlier works, like the 'Qualitative Physics based on Confluences' or the 'Qualitative Process Theory' are known to be quite limited in their abilities. Nevertheless, they had to be considered in this overview because they have largely influenced the other approaches and are therefore essential for the understanding of the still ongoing research.

This cross-section of relevant qualitative modelling techniques showed that among all *causal* approaches, only the artificial intelligence approaches Mycroft and QSIM with its latest extensions, as well as the largely overlooked system science-rooted Three-Layer-Model are applicable for modelling and simulation in control engineering. Overall, however, fuzzy modelling got the best ratings in the different categories of this comparison. Based on Zadeh's early papers from the late 1960's, fuzzy modelling is not only the most promising but also one of the oldest approaches dealing with qualitativeness. Still though, fuzzy modelling is an active field of research.

After the above analysis of qualitative modelling approaches and their shortcomings, the authors suggested an approach to improve the fuzzy-based modelling of dynamic processes by combining it with differential equations to a hybrid modelling concept [48], while the

improvement of aspect e) - the simplification of fuzzy-based qualitative modelling - is currently being investigated.

Despite the advantages of the fuzzy-based approach and its flexibility with respect to integrating other types of knowledge, it must not be considered as the 'single best' solution but it should be complemented by one of the prime *causal* approaches where purely *causal* qualitative modelling and simulation is required. Combining fuzzy based modelling with a *causal* qualitative modelling approach - either the Three-Layer-Model, Mycroft or QSIM with extensions - as well as conventional quantitative modelling into a single modelling approach is the overall aim of the author's current research [49]. This integrated modelling approach should eventually support engineers - even without specialist experience - in always applying the most appropriate model type, depending on the kind of knowledge available.

References

[1] J. Lunze, "Bemerkungen zur qualitativen Beschreibung dynamischer Systeme", *Automatisierungstechnik - at*, R. Oldenbourg Verlag, vol. 40, no. 8, pp. 281-284, 1992

[2] J. Lunze, "Ein Ansatz zur qualitativen Modellierung und Regelung dynamischer Systeme", *Automatisierungstechnik - at*, R. Oldenbourg Verlag, vol. 41, no. 12, pp. 451-460, 1993

[3] J. De Kleer and J. S. Brown, "A qualitative physics based on confluences", *Artificial Intelligence*, North-Holland Publishing Company, vol. 24, pp. 7-83, Dec., 1984

[4] K. D. Forbus, "Qualitative process theory", *Artificial Intelligence*, North-Holland Publishing Company, vol. 24, pp. 85-167, Dec., 1984

[5] P. A. Fishwick, "Fuzzy simulation: specifying and identifying qualitative models", *Int. J. General Systems*, vol. 19, pp. 295-316, 1991

[6] F. Puppe, *Systematic introduction to expert systems*, Springer-Verlag, 1993

[7] A. L. Stevens et al., "A program for handling multiple phases of the design cycle in process control system design", *Engng Applic. Artif. Intell.*, Pergamon Press, vol. 6, no. 3, pp. 231-239, 1993

[8] M. L. Mavrovouniotis and G. Stephanopoulos, "Formal order-of-magnitude reasoning in process engineering", *Comput. Chem. Engng.*, vol. 12, no. 9/10, pp. 867-880., 1988

[9] M. L. Mavrovouniotis, "A belief framework for order-of-magnitude reasoning", submitted for publication, 1994, please contact Prof. Dr. Mavrovouniotis at Northwestern University, Chem. Eng. Department, Evanston, IL 60208-3120.

[10] P. J. Hayes, "The naive physics manifesto", D. Michie (Ed.): *Expert systems in the microelectronic age*, Edinburgh University Press, 1978

[11] B. C. Williams, "MINIMA a symbolic approach to qualitative algebraic reasoning", *Proc. AAAI 88*, pp. 264-269, 1988

[12] R. Simmons, "'Commonsense' arithmetic reasoning", *Proc. AAAI 86*, pp. 118-124, 1986

[13] K. D. Forbus, "The qualitative process engine", in Weld and De Kleer (Eds.): *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufmann, pp. 220-235, 1990

[14] B. Falkenhainer and K. D. Forbus, "Compositional modelling of physical systems", *Recent advances in qualitative physics*, MIT Press, pp. 33-48, 1992

[15] K. D. Forbus and B. Falkenhainer, "Self-explanatory simulations: integrating qualitative and quantitative knowledge", *Recent advances in qualitative physics*, MIT Press, pp. 49-65, 1992

[16] B. D'Ambrosio, *Qualitative process theory using linguistic variables*, PhD-Thesis, University of California, Berkeley, 1986

[17] B. D'Ambrosio, *Qualitative process theory using linguistic variables: symbolic computing*, Springer-Verlag, 1989

[18] B. Kuipers, "Commonsense reasoning about causality: deriving behaviour from structure", *Artificial Intelligence*, North-Holland Publishing Company, vol. 24, pp. 169-203, Dec., 1984

[19] B. Kuipers, "Qualitative simulation as causal explanation", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-17, no. 3, 1987

[20] D. T. Dalle Molle, B. J. Kuipers and T. F. Edgar , "Qualitative modelling and simulation of dynamic systems", *Comput. Chem. Engng.*, Pergamon Press plc, vol. 12, no. 9/10, pp. 853-866, 1988

[21] B. Kuipers, "Qualitative simulation", *Artificial Intelligence*, Elsevier Science Publishers B.V. (North-Holland), vol. 29, pp. 289-338, 1986

[22] B. Kuipers, *Qualitative reasoning: modeling and simulation with incomplete knowledge*, MIT Press, 1994

[23] B. Kuipers and D. Berleant, "Using incomplete quantitative knowledge in qualitative reasoning", *Proc. AAAI-88*, pp. 324-329, 1988

[24] D. Berleant and B. Kuipers, "Qualitative-numeric simulation with Q3", *Recent advances in qualitative physics*, MIT Press, pp. 3-16, 1992

[25] H. Kay and B. Kuipers, "Numerical behavior envelopes for qualitative models", *Proc. of the 11th Nat. Conf. on Artificial Intelligence (AAAI-93)*, MIT Press, pp. 606-613, 1993

[26] A. Farquhar, *Automated modelling of physical systems in the presence of incomplete knowledge*, PhD Thesis, University of Texas at Austin, 1993

[27] J. Crawford, A. Farquhar and B. Kuipers, "QPC: A compiler from physical models into qualitative differential equations", *Recent advances in qualitative physics*, MIT Press, pp. 17-32, 1992

[28] A. Farquhar and G. Brajnik, "A semi-quantitative physics compiler", *Working Papers of the Int. Workshop on Qualitative Reasoning (QR-94)*, 1994

[29] O. Raiman, "Order of magnitude reasoning", *Proc. AAAI*, pp. 100-104, 1986

[30] Q. Shen and R. Leitch, "Fuzzy qualitative simulation", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 4, pp. 1038-1061, 1993

[31] Q. Shen and R. Leitch, "Combining qualitative simulation and fuzzy sets", *Recent advances in qualitative physics*, MIT Press, pp. 83-100, 1992

[32] G. M. Coghill and M. J. Chantler, "Mycroft: a framework for qualitative reasoning", *Intelligent Systems Engineering*, IEE Conference Publ. No. 395, pp. 43-48, Sept. 1994

[33], L. A. Zadeh, "Fuzzy sets", *Info and Control*, no. 8, pp. 338-353, 1965

[34] J. E. Hopcroft and J.D. *Ullmann, Introduction to automata theory, languages and computation*, Addison-Wesley Publishing Company, 1979

[35] A. Bredebusch, J. Lunze and H. Richter, "A Petri-Net representation of the qualitative behaviour of a dynamical continuous-time system", *Intelligent Systems Engineering*, IEE Conference Publ. No. 395, pp. 223-228, Sept. 1994

[36] Lunze, J, "A Petri-net approach to qualitative modelling of continuous dynamical systems", *SAMS*, Gordon and Breach Science Publishers, vol. 9, pp. 89-111, 1992

[37] M. Kluwe, V. Krebs, J. Lunze and H. Richter, "Beratungssystem fuer die operative Prozessfuehrung", *Automatisierungstechnische Praxis atp*, vol. 36, no. 8, pp. 11-16, 1994

[38] V. Krebs, M. Kluwe, J. Lunze and H. Richter, "Ein hybrides Modell zur qualitativen und quantitativen Beschreibung komplexer dynamischer Systeme", *Berichte vom 39. Internationalen Wissenschaftlichen Kolloquium in Ilmenau*, Verlag: Universitaetsbibliothek der TU Ilmenau, vol. 3, pp. 159-166, 1994

[39] M. Kluwe, V. Krebs, J. Lunze and H. Richter, "Qualitative modelling based on rules, Petri-nets, and differential equations", *Proc. on IMACS Symposium on Mathematical Modelling 1. Mathmod*, Vienna, Technical University Vienna, vol. 1, pp. 134-137, 1994

[40] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling", *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, Feb., 1993

[41] D. T. Dalle Molle, *Qualitative simulation of dynamic chemical processes*, PhD-Thesis, University at Austin, TX, 1989

[42] B. C. Williams, "Doing time: putting qualitative reasoning on firmer ground", *AAAI-86 Conference*, Philadelphia PA, pp. 105-112, 1986

[43] H. Ushida, T. Yamaguchi, K. Goto and T. Takagi, "Fuzzy-neuro control using associative memories, and its applications", *Control Eng. Practice*, Pergamon Press Ltd, vol. 2, no. 1, pp. 129-145, 1994

[44] K. Goto, T. Yamaguchi and T. Takagi, "Dynamic model for a plant using associative memory system", *Artificial Neural Networks*, 2 - Proc. of the '92 Int. Conf. (ICANN-92), Elsevier Science Publishers B.V., pp. 1517-1520, 1992

[45] M. Sugeno and G. T. Kang, "Fuzzy modelling and control of multilayer incinerator", *Fuzzy Sets and Systems*, Elsevier Publishers B.V., vol. 18, pp. 329-345, 1986

[46] R. Rajagopalan, "Qualitative modelling and simulation: A survey", *AI applied to Simulation*, Proc. of Europ. Conf., pp. 9-26, 1986

[47] D. A. Linkens, "AI in control systems engineering", *The Knowledge Engineering Review*, no. 5:3, pp. 181-214, 1990

[48] M. Strickrodt and K. Baker, "A fuzzy hybrid model for the simulation of nonlinear dynamic processes", currently under review for the IEEE Transactions on Systems, Man and Cybernetics.

[49] M. Strickrodt, R. Schumann and K. Baker, "An integrated knowledge engineering approach to process modelling for CACSD", submitted for the IFAC'96 World Congress.